PERSONALNI

računari

86

NOVEMBAR 1992.

CENA 2300 D

SLT 500 • Denara 4600 • DEM 10 • USD 6 • ATS 70



[illegible] DODATAK NA 32 [illegible]

IMAM RAČUNAR - ODAKLE DA POČNEM?

HARDVER

INTE[illegible]NI DISK [illegible]OLERI

PROGRAMSKI JEZICI

SA C na C++ U PET [illegible]

SOFTVER

TURBO C++ 3.0 FOR WINDOWS

BORLAND PARADOX 4.0

AUTODESK AUTOCAD 12

QUATTRO PRO FOR WINDOWS

NOVELL LITE

TIPOTEK: FONTOVI [illegible]

NiCBASE: ELEKTRONSK[illegible]

AWK: JEZIK ZA SVAKI DAN

MIKROPRO[illegible]ORI

P586



PERSONALNI
računari 86

časopis za informatiku i računarstvo
YU ISSN 0352-7221
Izdaje BIGZ
novembar 1992 cena 2300 din

Izdaje i štampa
Beogradski izdavačko grafički zavod
11000 Beograd
Bulevar Vojvode Mišića 17

Generalni direktor
Ilija Rapajić

Direktor Novinskog sektora
Manja Ajtić

Glavni i odgovorni urednik
Jovan Regasek

Komercijalni urednik
Vesna Jeremić

Tehnički urednik
Radoje Kavedžić

Urednik za hardver
Dejan V Veselinović

Stručna redakcija
Nenad Balaćanin (baze podataka) Ranka Jovanović (aktuelnosti) Dejan Ristanović (softver) Jovan Skuljan (programiranje) Vladimir Stamenović (radne tabele) Dejan Veselinović (hardver) Dragutin Vuković (mreže) Zoran Živolić (softver)

Adresa redakcije
11000 Beograd
Bulevar vojvode Mišića 17/III

Telefoni
653 748 (redakcija) (prodaja) 651-666 i 259

Agencija BIGZ-a
(011) 651-793 653-565

Telefaks
(011) 648-140 647-955

Pretplata za zemlju
Za šest meseci zvati prodaju
Za jednu godinu zvati prodaju
(na žiro račun D P BIGZ 60802-603-23264)

Pretplata za inostranstvo
Za jednu godinu 51 USD 80 DEM 67 CHS 27 GBP 271 FRF
(na devizni račun D P BIGZ 60811-620-16101 820701-999-03377)
Zbog usporenog prenošenja uplata preko banke molimo pretplatnike da nam posle svake nove uplate odmah pošalju foto-kopiju uplatnice
Rukopisi se ne vraćaju

SEZAM (011) 648-899 (deset linija)
sistem za modemske komunikacije
Upravnik sistema
Jovan Regasek
Administratori na sistemu
Zoran Životić i Dejan Ristanović

SADRŽAJ



Na naslovnoj strani Senja Obradović, 21 godina, student menadžmenta, maneken i foto-model. Fotografija. Studio Tešić & Nenad Petrović.

SADRŽAJ OGLASA

ŠTA IMA NOVO

TRŽIŠTE

Prodaja

Najveći prihodi od Windows-a

U drugom kvartalu 1992. godine prodaja PC softvera je porasla za 20%, dosevši 1,49 milijardi dolara, pri čemu najveći udeo ostvaruje softver za *Windows* operativne sisteme.

U proleće ove godine prodaja aplikacija za *Windows* je bila za čitavih 181% veća od prošlogodišnje i iznosila je 526 miliona dolara. U isto vreme, prodaja MS-DOS aplikacija je opala za 13% i iznosila je 637 miliona dolara.

Transferi

Windows NT za korisnike radnih stanica

Prema rezultatima ankete koju je sproveo UNIX-ov list „Open Systems Today", nekadašnji „Unix Today", većina korisnika UNIX-ovih radnih stanica koji sada rade sa UNIX-om na RISC platformi želi da pređe na *Microsoft Windows NT* operativni sistem. Naime, od korisnika je traženo da ocene šest klijent-server operativnih okruženja sledeće generacije i 63% ispitanika je stavilo *Windows NT* na prvo mesto, dok je na drugom mestu Solaris.

OPERATIVNI SISTEMI

Windows for Workgroups

Nova Microsoft izmišljotina: Windows za radne grupe

U besomučnoj trci za prevlast u svetu računara „Microsoft" je opet smislio nešto novo: krajem septembra predstavljen je operativni sistem *Windows for Workgroups*.

Bil Gejts objašnjava ovaj novi korak: „Svedoci smo da se u svetu polako napušta rad na usamljenim ostrvima informacija, u zasebnim radnim stanicama, i da se sve više ide na grupe ljudi koji zajednički rade na istom poslu koristeći iste informacije. „Microsoft" je stvorio operativni sistem za razvoj nove generacije aplikacija: one će služiti ovakvim radnim grupama i u isto vreme ostvaruju naše vizije o informacijama koje će se, pritiskom na taster, naći na svakom radnom stolu."

Dakle, *Windows for Workgroups* je specijalna verzija programa *Microsoft Windows 3.1* namenjena grupama PC-korisnika koji rade na istim poslovima i koriste zajedničke informacije. Program ima ugrađenu elektronsku poštu, grupno planiranje i dinamičnu razmenu podataka (DDE) za rad u mreži. Konačna verzija će imati softver za mrežnog klijenta kompatibilan sa *Novell NetWare* i *Microsoft LAN Manager*-om, kao i usavršen *Windows File Manager*, koji ga čini najboljim *Windows* klijentom za ove mreže.

Jedna od odlika „Microsoft"-ove nove strategije je otvorenost kompjuterskog okruženja za sve potrebe korisnika i programera. Logičan pristup za ovakve funkcije je *Windows Open Services Architecture* (WOSA), arhitektura koja omogućava korišćenje velikog broja novih opcija preko već poznatog interfejsa. Istu fleksibilnost WOSA pruža i programerima, koji se mogu potpuno posvetiti razvoju inovativnih rešenja bez potrebe da programiraju dodatne funkcije na nivou sistema. Dakle, i korisnici i programeri mogu kombinovati i spajati razne proizvode i opcije da bi došli do željenog rešenja.

„Microsoft" se pobrinuo i za integrisanje operativnih sistema *Windows for Workgroups* i *Microsoft Windows NT* u postojeće mrežne sisteme, kao i za upravljanje budućim sistemima u velikim firmama. Za preduzeća čija se računarska infrastruktura sastoji od *host*-ili integrisanih sistema, WOSA omogućava lak prelazak na LAN klijent-server rešenja.

Predviđeno je prilagođavanje postojećih aplikacija, kao što su *Microsoft Excel* i *Microsoft Word*, radu u radnim grupama preko standardnih API-ja (na primer MAPI), kao i kreiranje novih aplikacija i tehnologija neophodnih kako pojedincima tako i grupama.

Razvoj softvera koji će predstavljati potpuno rešenje za jednu radnu grupu zahteva složenu infrastrukturu za servisiranje, podršku i stručnu ekspertizu. Sa svojim nedavno uspostavljenim planom podrške nazvanim *Solutions Channel*, „Microsoft" nudi programerima i integratorima direktan pristup potrebnim alatkama, podršci i tehničkim informacijama.

MS DOS

Verzija 6.0 od septembra na beta-testu

Nova verzija MS-DOS-a donosi novine pre svega u pomoćnim programima i radu u mrežama. „Microsoft" je preuzeo najbolje stvari iz različitih pomoćnih programa i ugradio ih u svoj operativni sistem. Čak su preuzeti i neki moduli iz Nortona. Tako će delovi *Norton Backup*-a zameniti dosadašnju DOS naredbu **Backup**. Zbog svega toga proizvođači softverskih alatki, kao što je Fifth Generation, ne gledaju na „Microsoft" sa odobravanjem jer ugrožava njihov deo tržišta.

Najvidljivije promene MS-DOS 6.0 doživeće pre svega u radu sa mrežama. „Microsoft" se ne trudi da poboljša i unapredi samo svoje specijalne mrežne proizvode, kao što su *Windows for Workgroups*, *Windows NT*, *LAN Manager* i *LAN Server*, već bi nova verzija MS-DOS-a trebalo da bude upotrebljiva i za rad sa drugim mrežama. „Microsoft" sada pregovara sa „Novell"-om da MS-DOS 6.0 radi i u njihovoj mreži. Ovo mrežno povezivanje bi omogućilo onima koji rade u DOS-u da što jednostavnije koriste pogodnosti rada u mreži. Ako DOS korisniku, na primer, zatreba u mreži *Windows for Workgroups*, onda mu treba samo mrežna kartica sa kablovima i već je spreman da aktivira neku grupno orijentisanu aplikaciju. Uz sve ovo, MS-DOS 6.0 bi trebalo da korisniku oslobodi dodatnu količinu radne memorije.

Pen OS

Ratovi za OS pen računara: DOS protiv Windows-a

Dok se na poprištu pen računara vodi ljuti boj za operativni sistem koji će imati prevlast, praktičnost pen računara kao alatki za prikupljanje podataka usmerava tržište ka sistemima koji su kompatibilni sa onim što korisnici već imaju.

Na izložbi pen računara (Pen-based Expo) održanoj krajem septembra u Kaliforniji, sve su preplavili *Windows* sistemi i svi su izgledi da će „Microsoft" iz rata za operativni sistem izaći kao pobednik. Među tim, „Microsoft"-ov *Windows for Pen Computing* vapi za hardverom, kao i *Microsoft Windows 3.1*, pa neki dileri pokušavaju da istaknu prednosti DOS pen operativnih sistema.

Jedan od sistema u toj igri je i *PenDOS*. „Communication Intelligence Corporation" (CIC), firma koja je razvila *PenDos*, ističe da ovaj operativni sistem zahteva samo 2 megabajta RAM-a, dok je za *Windows* potrebno 6 – 8 MB. *PenDOS* je zasnovan na verziji 3.0 i radi sa svim DOS aplikacijama, pa i onima koje nisu razvijene za pen sisteme. Demonstriran je rad u *Microsoft Windows*-u i *Lotus*-u 1-2-3 pod PenDOS-om na NCR pen računaru, a predstavljene su i DOS aplikacije koje su sasvim lične na *Windows*.

Penpoint operativni sistem firme „Go Technology" je bio mnogo manje primetan od *Windows*-a. Mada se govorkalo o konfliktu između dva proizvođača, izgleda da se ipak radi o dva različita usmerenja. „Go" je svoj operativni sistem namenio specijalizovanom vertikalnom tržištu aplikacija koje ne zahtevaju čvrst interfejs između desktop računara i pen jedinica, dok Microsoft i CIC naglasak stavljaju na kompatibilnost.

Ako postoji rat, onda je to onaj koji se vodi između DOS-a i *Windows*-a. *PenDOS* ističe svoju prednost: korisnik mora naučiti samo sedam osnovnih poteza, dok je za *Pen Computing Windows* potrebno trinaest.

Međutim, niko se još posebno ne bavi raspoznavanjem rukopisa (HWR), uglavnom zato što ono uveliko i dalje zavisi od urednog, što znači sporog, ispisivanja. Naglasak je stavljen na prikupljanje podataka s obzirom da pen sistem predstavlja jednostavan i prirodan način za elektronsko popunjavanje formulara.

PenDOS procenjuje da bi jedna kompanija za osiguranje koja bi koristila aplikaciju za popunjavanje formulara uštedela oko 1500 sati mesečno. Toliko vremena provode službenici koji podatke sa rukom pisanih formulara unose u računar. Osim toga, PenDOS tvrdi da njihov sistem prihvata i interpretira unos brže od *Windows*-a, pošto je DOS praktičniji. No, uprkos svemu ovome, svi su izgledi da će pobedu ipak odneti – „Microsoft".

SOFTVER

Zaštite

Uskoro zakon o kopirajtu softvera u Rusiji

Zakon o kopirajtu softvera i baza podataka konačno ima šansu da se probije kroz lavirint ruske zakonodavne birokratije. Naime, prošao je sve faze pripreme i, nakon što je dva puta razmatran u Vrhovnom Sovjetu, prosleđen je ruskom predsedniku Jeljcinu. Međutim, Jeljcin je odbio da ga potpiše. Jedan od razloga je rupa koja postoji u ruskim saveznim zakonima, po kojoj je intelektualna svojina zajednička svojina Federacije i njenih članica (autonomnih pokrajina). To znači da zakon prvo treba da odobre sve republike unutar ruske Federacije.

Nesuglasice između onih koji su za ili protiv novog zakona odražavaju se na stranicama ruske štampe. Udruženje softverskih dilera tvrdi da Ruska agencija za intelektualnu svojinu (RAIS) stvara prepreke i pokušava da uspori donošenje zakona, koji inače treba da zaštiti one koji se bave prodajom softvera. RAIS, opet, u potpunosti odbija ove tvrdnje.

Zakon će se uskoro po treći put naći na dnevnom redu ruskog parlamenta. Ako tada bude usvojen, više neće biti potreban Jeljcinov potpis da bi počeo da se primenjuje.

Stono izdavaštvo

Quark Xpress za Windows – sanjali ste, gledajte



Ljubimac Macintosh korisnika, DTP program *Quark Xpress* zakoračio je i u *Windows* svet i trenutno je na raspolaganju u beta-verziji. Na prvi pogled se vidi da program vodi direktno poreklo od programa *Quark Xpress 3.1* za Macintosh, jer su interfejs prema korisniku i opseg funkcija do najmanjih detalja podudarni sa Mac verzijom. Mada se bez problema instalira na 386 mašini, za optimalno iskorišćenje programa preporučuje se 486 mašina. *Quark Xpress for Windows* donosi brojne mogućnosti za prelom i oblikovanje strana, kojih su do sada DTP korisnici koji su radili pod *Windows*-om morali da se odreknu. Tu se ubrajaju funkcije za precizno pozicioniranje teksta i grafike, opsežne tipografske mogućnosti, rotacija teksta i kompletna paleta boja. Sada će DTP pod *Windows*-om i za profesionalne korisnike postati interesantan. Smatra se da će *Quark Xpress for Windows* u budućnosti igrati važnu ulogu na *Windows* tržištu. Već je i beta verzija ostavila solidan utisak. Trebalo bi da cena ovog programa bude u skladu sa cenom za Macintosh-verziju i da se kreće negde oko 2500 DEM.

Baze podataka

FoxPro za Windows i još tri verzije

„Microsoft" je najavio četiri nove verzije programa FoxPro. FoxPro 2.5 i FoxPro for Windows se očekuju krajem godine, dok će se FoxPro za Macintosh i Unix pojaviti u prvoj polovini 1993.

FoxPro for Windows koristi prednost Windows grafičkog *okruženja* i omogućava upotrebu različitih fontova i novih ikona. *Windows* (kojeg prati glas da usporava aplikacije) neće degradirati već poznate visoke performanse programa *FoxPro* jer *FoxPro* mašina ostaje u osnovi ista. Prilikom nedavne demonstracije prikazana je *FoxPro for Windows* baza podataka u koju su unesene sve ulice u SAD (oko 2.6 miliona) – program je odradio nekoliko pretraživanja (**search** operacije) za samo dve-tri sekunde.

Za razvijače softvera veoma je značajno to što *FoxPro for Windows* omogućava izvršavanje *FoxPro* programa pisanih za MS-DOS bez izmena. Ima ugrađen konvertor koji automatski prevodi *FoxPro 2.x* programe u *FoxPro for Windows*, a softveraši mogu ići i dalje od toga uz pomoć specijalnih *Windows* alatki.

Ono što *FoxPro for Windows* čini posebnim je način na koji koristi tri osnovne *Windows* komponente DLL (*dynamic link libraries*), DDE (*dynamic data exchange*) i OLE (*object linking and embedding*). DDE je demonstriran sledećim primerom: sve funkcije tekst procesora *Word for Windows* se mogu dodati programu *FoxPro*. Iako što *FoxPro* zahteva da *Word* obavi mail-merge, formatira dokumenta i pripremi „mail-merge" izlaz. Alternativa je da *FoxPro* obavi ulogu „servera" za *Word* („klijent"), pretvarajući *Microsoft Word* u pretprocesor za brzu i moćnu database mašinu.

Demonstracija je krunisana *FoxPro Windows* aplikacijom u koju su uz pomoć OLE ugrađene slike, video animacije i glasovne poruke, čime je *FoxPro* pretvoren u multimedijalnu aplikaciju.

Razvojni alati

C++ za Windows bez po muke

Oni koji se bave razvojem *Windows* softvera i koriste *KnowledgePro for Windows* sada mogu da razvijaju C++ programe pomoću alatke koja se zove *KPWin++ Windows Development Tool*. To praktično znači da možete praviti C++ prototipske aplikacije koristeći *KPWin* interaktivne alatke za projektovanje kao i njihov jezik. *KPWin++* koristi sve sposobnosti KPWin jezika – ekspertni sistemi, hipertekst, multimedija, rad s listama i datotekama.

Ova alatka može da čita kod pisan u *KPWin* ili *OpenInsight* okruženju, a generiše C++ kod koji se prevodi ANSI kompajlerom. Uz pomoć *Microsoft C/C++ 7.0* kompajlera moguća je dorada generisanog C++ koda, povezivanje sa bibliotekama nezavisnih proizvođača i ponovno kompajliranje da bi se proizvela izvršna datoteka visokih performansi.

Kontakt adresa: Knowledge Garden Inc., 12-8 Technology Dr., Setauket NY 11733 (516) 246-5400, fax (516) 246-5452

Multimedija

Sound Impression: editovanje zvuka za Windows

„Digivox" predstavlja *Sound Impression*, novi desktop 16-kanalni rekorder za korisnike *Windows-a 3.1* koji omogućava snimanje zvuka, editovanje i miksovanje. Korisnički interfejs liči na komponentu stereo-sistema, sa već prepoznatljivim funkcijama. Na primer, jačina i balans tona se podešavaju pomoću miša. Korisnici mogu stvarati bilo koju vrstu zvuka, kao i složene 16-kanalne kompozicije. Talasni editor omogućava isecanje i lepljenje zvuka iz 16 različitih setova, a mono-talasne forme se mogu pretvarati u stereo forme i obratno. Tu je i mogućnost dodavanja posebnih efekata kao što su eho, fejding, pan, krosfejding. *Sound Impression* čine: naslovni panel (Title Panel), miksing panel, talasni rekorder/plejer i MIDI plejer. Naslovni panel sadrži dugmad i opcioni meni tastera za upravljanje datotekama kao i druge funkcije. Miksing panel ima kontrolne tastere za podešavanje jačine i balansa tona za ulaz uređaja za snimanje, talasnih datoteka, digitalnog interfejsa muzičkih instrumenata (MIDI) i kompakt diskova, kao i master kontrolu za jačinu i balans glavnog uređaja. Ulaz se vrši preko mikrofona ili preko pomoćnog inputa.

Talasni rekorder/plejer snima i reprodukuje talasne datoteke, nudi **lup** funkciju za neprekidno reprodukovanje i omogućava pristup onim delovima programa u kojima se vrši editovanje i komponovanje. MIDI plejer reprodukuje MIDI datoteke, dok CD plejer može reprodukovati CD sa CD ROM-a.

Po računu proizvođača, korisnici mogu do mile volje menjati efekte. Na primer, **eho** funkcija ima tri nivoa i za svaki se može posebno odrediti interval, a tu je i mogućnost kanonskog ili horskog efekta.

Ova sprava zahteva 386 ili moćniji IBM-kompatibilni PC računar sa najmanje 4 MB memorije, ali se preporučuje 8 MB. Potreban je hard-disk sa 2 MB raspoloživog prostora, kao i zvučna kartica sa *Windows 3.1* multimedija drajverima, dok CD ROM nije obavezan. Cena će biti oko $149.

TEHNOLOGIJA

Standardi

Da li ćemo dobiti poboljšanu EISA magistralu?

Izvestan broj firmi koje su učestvovale u stvaranju EISA magistrale ozbiljno razmišlja o njenoj novoj verziji. Inženjeri firmi AST, „Corollary", „Digital Equipment", „Distributed Processing Technology" (DTP), „Intel" i „Wyse" planiraju proširenje EISA magistrale na 64 bita uz veću propustljivost podataka. Razloga za to ima više, a prvi je to što se procenjuje da će Multimedija i 32-bitni operativni sistemi „prerasti" kapacitet EISA magistrale već za godinu ili dve dana. Drugi razlog je to da IBM vrlo izvesno radi na novom Mikrokanalu, pa konkurenciju treba preduhitriti.

RAČUNARI

Računari

Compaqovi novi serveri: brži, bolji, jeftiniji

„Compaq" najavljuje da će svojim novim proizvodima *ProSignia* i *Systempro/XL* iz temelja potresti tržište servera, baš kao što je u junu potresao desktop i *notebook* tržište. *ProSignia* je 486 33 MHz žaji server koji je za 64% brži i 2.200 dolara jeftiniji od „Dell"-ovih 466/50 servera, a 1.200 dolara jeftiniji od 486sx/33 IBM-ovog PS/2 Modela 85. Cene najjeftinijih *ProSignia* servera nešto su niže od 4.500 USD.

„Compaq" tvrdi da je *Systempro/XL* najbrži PC server na svetu (cene se kreću od 14.000 USD naviše), 89% brži od IBM-ovog modela 95/XP koji se prodaje po gotovo istoj ceni. Novi serveri će imati ugrađen *NetWare for Compaq*, specijalnu verziju „Novell"-ovog mrežnog operativnog sistema. Tu je i *Insight Manager Windows* alatka za upravljanje celokupnom mrežom Compaq PC servera.

Odnedavno „Compaq" daje trogodišnju garanciju, a uvedena je i 24-časovna telefonska služba za rešavanje hardverskih problema kod novog servera, kao i dve službe koje daju savete u slučaju problema sa radom u mreži.

Još jedan novitet u „Compaq"-ovoj liniji hardvera je *ProLiant* SCSI-2 eksterni sistem za skladištenje koji prima više od 7 GB podataka. Spoljno skladištenje će omogućiti *Netware* korisnicima da zamene drajv koji dobro ne radi bez obaranja mreže. Jedna mreža može koristiti četiri *ProLiant* uređaja, a obezbeđena je podrška za *Compaq Deskpro/M*, *Systempro/LT* i *Systempro*. ProLiant ima SCSI-2 kontroler, 32-bitni kontroler mrežnog interfejsa i 1024x768 grafički kontroler. Procesorski čip se može dograditi zamenom postojećeg 33 MHz Intel 486 nekim Intel-ovim Overdrive procesorom, na primer sa 486DX2/66.

Systempro/XL se očekuje krajem oktobra, dok se većina modela *ProSignia* već prodaje širom sveta.

Notebook

IBM na popravnom ispitu: ThinkPad i njegovo društvo

IBM je izbacio na tržište liniju *ThinkPad* "notebook" računara sa tri nova modela tastature, kao i zvaničnu verziju pen *ThinkPad* računara, snizivši u isto vreme cene ranijih portabl modela.

Predstavnici IBM-a tvrde da će nove mašine popraviti dosta slabu reputaciju koju su IBM prenosnici do sada uživali. Jedan od novih modela – *ThinkPad 700C* ima veliki kolor ekran, a tu su i dve monohromatske jedinice nazvane *ThinkPad 300* i *ThinkPad 700*. Pen model je dobio novo ime: *ThinkPad 700T*.

ThinkPad 700C ima ekran dijagonale od 10.4 inča rađen u tehnologiji aktivnih matričnih tankoslojnih tranzistora. Proizvodi ga „Display Technologies", firma koju su nemenski osnovali IBM i Toshiba da bi ih snabdevala kvalitetnim ekranima.

Još jedna inovacija koju donose modeli *ThinkPad 700* i *700C* je pokazivač (pointer) smešten u sredinu tastature. Pokazivač *TrackPoint II* je osetljiv na dodir i pokazuje u svim smerovima, a nalazi se između tastera za „G" i „H" – na zgodnom mestu i za dešnjake i za levake. Tako ruke korisnika sve vreme ostaju na tastaturi, što je praktično naročito kada se radi u skučenom prostoru – na primer u avionu. *TrackPoint* ne oduzima puno mesta na tastaturi i neće stvarati probleme daktilografima koji kucaju „naslepo".

IBM ThinkPad 7000C je zasnovan na IBM 486SCL/25*MHz* procesoru koji se može dograditi na 50 MHz. Ima prenosni hard-disk od 120 MB, 4 MB memorije koja se može proširiti do 16 MB, IBM-ovu *Micro Channel Architecture* sistemsku sabirnicu, koprocesorsko podnožje i bateriju od nikl hidrida koja sa jednim punjenjem omogućava 3.8 sati rada. 700C teži 3.5 kg i koštaće oko 4.350 USD.

ThinkPad 700 je monohromna verzija modela 700C, nešto lakša od 3 kg, sa ekranom od 9.5 inča, a može se dograditi tako da radi sa aktivnim matričnim displejem. Sa hard-diskom od 80 MB košta $ 2.750, dok model sa diskom od 120 MB košta $ 2.950.

„Low-end" model nove linije je *ThinkPad 300* sa monohromatskim ekranom i 25 MHz 386SL procesorom. Kapacitet hard-diska se može proširiti sa 80 na 120 MB, dok se dogradnjom crno-beli ekran može zameniti onim u boji. Baterija omogućava rad od 4-10 sati, zavisno od konfiguracije sistema. Ugrađena je podrška za „Ethernet" lokalne mreže.

IBM nudi i port replikator za *ThinkPad 300* koji se može priključiti na periferije, mreže i druge komunikacione uređaje, s tim da sve ostaje na svom mestu kada krenete na put sa svojim „portablom". *ThinkPad 300* sa drajvom od 80 MB će koštati oko $ 2.375, a model sa 120 MB oko $ 2.575.

Originalni pen *ThinkPad* model sada je nazvan *700T*. Rađen je sa *PenPoint* operativnim sistemom preuzetim od korporacije „Go", dok je tehnologija za raspoznavanje rukopisa razvijena u IBM-u. Ima 20 MHz 386SX mikroprocesor, između 4 ili 8 MB memorije, integrisan *data/fax* modem, AC/DC adapter, nikl kadmijum bateriju, serijske i paralelne portove, priključak za eksterni disketni drajv od 3.5 inča, kao i eksterni priključak za tastaturu. IBM je unapredio i prepoznavanje rukopisa. *700T* sa 4 MB memorije će koštati oko $ 4.795, a sa 8 MB oko $ 5.395.

KARTICE

Video

Nove grafičke kartice

„ATI Technologies" je predstavio dve nove brze grafičke kartice pod nazivima *Graphics Ultra+* i *Graphics Ultra Pro*. One bi trebalo da objedine veliku brzinu, bogatstvo boja i visoku rezoluciju. U kolor režimu dozvoljavaju prikaz 65000 boja pri rezoluciji od 1024x768 tačaka, dok u režimu bez preplitanja omogućavaju prikaz 256 boja pri rezoluciji od 1280x1024



Samo je jedan
Olymp
elektronic
PC - AT
286
386
486
PRINTERI
LX-400 LQ-870
LQ-200 LQ-1070
LQ-570 LQ-1170
CITIZEN GSX-140 COLOR
HP LASER JET III
HP LASER JET IIIP
HP DESK JET 500C
Računarske
mreže
NOVELL
PRATEĆA
OPREMA
11000 Beograd Jovana Đaje 10 tel 400-477 fax 410-240



tačke. Kartice će biti ponuđene u EISA, ISA i *MicroChannel* verzijama i podržavaju *Multimediju* i *AVI Playback*. *Graphics Ultra Pro* sa 2 MB memorije bi trebalo da ima cenu od 1495 DEM, dok za Ultra+ verziju cena u DEM još nije konačna, ali se za SAD kreće oko 399$.

Multimedija

Most između PC i videa u realnom vremenu

Uz pomoć *PV100 Video Window Adapter*a moguće je gledati „žive" video snimke pod DOS-om i Windows-om 3.x na PC-ju u realnom vremenu i sa kvalitetnim bojama. Ovaj sistem raspolaže sa tri video ulaza i može se lako softverski prebacivati između PAL i NTSC standarda. Video rekorder, video kamera i TV prijemnik se priključuju veoma jednostavno. Digitalizovane slike mogu se animirati u MMP, TGA ili BMP formatu, a uz pomoć programa *photo* koji se dobija uz adapter, animirane slike mogu se konvertovati u bilo koji drugi uobičajeni format. Digitalizator koji radi u realnom vremenu košta sam oko 1980 DEM, dok S-Video Kit koji ga obuhvata košta 2380 DEM. Za računarske animacije, prodajne i promotivne video snimke nudi se PV110 VGA to Video konvertor. Ova kartica sadrži „Tsengov" čip ET 4000, 1 MB video RAM, a rezoluciju od 1024x768 tačaka i sledeće izlaze: VGA, RGB, Video (FBAS) i S-Video. PV110 se prodaje u dve varijante: kao PV110-P sa 256 boja (1198 DEM) i kao PV110-PH sa 32768 boja (1498 DEM).

Multimedija

Kompresor za slike i digitalizovani zvuk

Optibase bi trebao da omogući korišćenje multimedija, što je do sada bilo rezervisano samo za radne stanice. Kartica za IBM-kompatibilna PC je za kompresiju slika i digitalizovanog zvuka obezbeđuje alatke uz pomoć kojih je moguće kombinovati visoko-kvalitetne slike u boji sa govorom i muzikom na PC-ju. Kompresija slika se obavlja JPEG, a audio kompresija ADPCM postupkom. „Techex" (8028 Taufkirchen) nudi dve verzije ovog proizvoda: *OP 100* koristi *Motorola 56001* procesor signala (1899 DEM), dok *OP 500* upošljava dva paralelna 56001 procesora (3499 DEM).

PERIFERIJE

Matrični štampači

Citizen: novi Swift-modeli

Uz moto „inovativna oprema i atraktivni dodaci" „Citizen" želi da svoju drugu generaciju modela matričnih štampača iz serije *Swift* uvede na profesionalno tržište. Pored poznatih funkcija kao što su parkiranja papira (*paper parking*) i skrolovanje papira (*paper scroll*), novi *Swift* modeli – 200, 240 i 240C – omogućavaju „četvrtinsku" štampu (*quarter printing*). Uz pomoć ove funkcije moguće je individualno oblikovana formulare: na primer umesto nekog rokovnika, odštampati četiri pula jedan uz drugi na stranici A4 formata. **Auto-Set** funkcija utvrđuje potrebnu emulaciju i parametre na koje je podešen računar i automatski konfiguriše štampač. Smanjen je i nivo buke koju prave ovi štampači i sada iznosi 47,4 dB(A). Maksimalna brzina štampanja modela 240 i 240C je 240 znakova u sekundi. Štampači će se standardno isporučivati sa devet fontova koji se mogu skalirati od 2 do 40 tačaka. Opciono se može dodati daljih šest fontova. Model *Swift 240* košta 946 DEM, a za dodatnih 139 DEM dobija se mogućnost štampanja u boji. *Model 240C* prodavaća se za 998 DEM. Model *Swift 200* „Citizen" nudi za 699 DEM. Ovaj poslednji štampa maksimalnom brzinom od 216 znakova u sekundi i isporučuje se sa sedam ugrađenih fontova. Za sva tri modela „Citizen" daje dvogodišnju garanciju.

Strimeri

Skladištenje podataka brzinom munje

„Exabyte" je predstavio EXB-8505, 8mm poluvisinski strimer od 5.25 inča, sa kapacitetom od 5 gigabajta i sposobnošću da prenos podataka obavlja brzinom od 500 kilobajta u sekundi. To znači da se bekap hard diskova od 40 MB završava za minut i po. Drajv omogućava kompresiju podataka kojom se kapaciteti i transfer mogu udvostručiti. Srednje vreme pojave greške (*mean time between failures* MTBF) inače industrijski standard za određivanje pouzdanosti hardvera, je 80.000 sati. Ovaj novi podsistem koristi IBM-ov algoritam kompresije (*Improved Data recording Capability* – IDRC), koji je ugrađen u čip projektovan u „Exabyte"-u. Proporcije kompresije zavise od podataka i aplikacije. „Exabyte" tvrdi da su testovi brzine na podacima korisnika davali odnose kompresije od 2:1.

Kopir aparati

Canon staje na put štampačima lažnog novca

Najnoviji model „Canon"-ovog kolor fotokopir aparata ima specijalni čip u čijoj memoriji je zabeležen izgled novčanica najvažnijih svetskih moneta. Kada neko pokuša da fotokopira novčanicu od jednog dolara, „Canon"-ov fotokopir je prepoznaje i izbacuje potpuno crni list hartije. Ugrađena je još jedna vrsta zaštite – svaka kopija se obeležava šifrom koja sadrži broj aparata i datum. Ideja o ugrađenoj zaštiti je potekla od policije i banaka koje muku muče sa falsifikatorima širom jugoistočne Azije i Evrope. Na razvoj nove tehnologije utrošeno je pet godina i „Canon", koji pokriva 90% japanskog tržišta, je spreman da je ustupi i drugim proizvođačima.

Tastature

Ergonomska tastatura – manji zamor

Mini Ergo-MF2 naziv je ergonomske tastature firme „Marquardt" (Nemačka). Prema podacima proizvođača, tastatura je bolje ergonomski oblikovana u odnosu na uobičajene tastature (što se i na slici jasno vidi) i trebalo bi da omogući rad bez zamora. Numerička tastatura doduše nije ugrađena, ali se može dokupiti. Mini-Ergo se direktno priključuje na sve AT kompatibilne računare, dok za PS/2 sisteme „Marquardt" nudi adapterski kabl. Cena ove tastature je 216 DEM.

ZANIMLJIVOSTI

Moda

Tašne za prenosnike italijanskog dizajna

Prošla su ona vremena kada se prenosnik nosio pod rukom. Ko danas putuje sa svojim voljenim „mališom", pakuje ga u kožu. „OFD Computersystems" (6073 Egelsbach) nudi kožne tašne za prenosnike u svim veličinama, pogodne za nošenje u ruci ili okačene o rame. Koža i italijanski dizajn imaju naravno i svoju cenu: ona se kreće od 800 do 1000 DEM. Zaštitne tašne za *Toshiba* i *Canon* modele mogu se nabaviti za „tričavih" 250 DEM.

DOMAĆI SOFTVER

Komunikacije

Distribucija softvera na Novell mreže preko IBM hosta

Jedan od nedostataka distributivne obrade podataka je u tome što je potrebno nove verzije aplikativnog softvera instalirati na više sistema. Ovaj problem je do danas ostao nerešen u slučaju *Novell* mreža povezanih na IBM *mainframe*. *Novell* je najavio odgovarajući softver koji će se do kraja godine pojaviti na tržištu. Međutim, Elektronski računski centar "Mešovite banke PKB D.D." Beograd je razvio paket koji omogućava distribuciju *Novell* aplikativnog softvera na sva mreže povezane na IBM *mainframe*, i to sa jednog mesta.

Ovaj paket se sastoji od PC programa i CICS transakcije na *hostu* i zasnovan je na IBM interfejsu APPC LU 6.2. Da bi ova komunikacija bila moguća, PC program mora biti instaliran na svakoj *Novell* mreži u sistemu. Da bi se inovirao aplikativni softver i na udaljenim lokacijama, potrebno je samo da korisnik startuje PC program, što će omogućiti da ERC sa razvojne mreže preuzme potpunu kontrolu nad stanicom. Naime, ovaj PC program se različito ponaša u zavisnosti od toga da li je startovan sa razvojne mreže ili sa korisnička mreža. Na razvojnoj strani se nudi niz mogućnosti, od slanja i preuzimanja podataka, do slanja poruka, i što je najvažnije, do izvršavanja komandi na korisničkoj mreži. Prednost ovakvog sistema rada je da modem koji služi za komunikaciju sa *hostom* ujedno služi i za distribuciju softvera i druge intervencije na udaljenim mrežama korisnika.

Kontakt telefon: (011) 768-359 Milvojević Siniša, (011) 752 684 Koković Vladimir

Nagradna igra

Štampač Citizen dobio Aleksandar Nikolić iz Beograda

U prošlom broju „Računara", kao u i prethodna tri, na određenom mestu stavljen je, u nasumično izabranom primerku, tajni znak koji će vlasniku doneti na poklon 24 iglični štampač 124D poznate firme „Citizen". Znak koji srećnom vlasniku obeleženog primerka „Računara" broj 85 donosi matrični štampač „Citizen" 124D – diskretni potpis našeg te[…]čkog urednika – nalazi se na 29 strani u dnu srednjeg stupca. Otvorite, dakle, stranu 29 u prošlom broju i, ako ste imali sreće, izvolite u redakciju po svoj „Citizen" štampač.



Broj 84 sa skrivenim znakom (crtež leptira) nije odmakao daleko: kupljen je Beogradu u Knez Mihajlovoj kod Kalimegdana, a srećni dobitnik Citizen štampača je Aleksandar Nikolić, Marijane Gregoran 17, 11060 Beograd. Gospodin Nikolić je glavni inženjer u Fabrici hartije. „Računare" čita od prvog broja i već ima jedan Citizen štampač, al sa devet iglica.

Firma darodavac štampača „Citizen" MERKOM iz Kruševca na domaćem tržištu štampača sa „Citizenovim" programom pojavila se sredinom 1990. godine. Standardna ponuda se sastoji od 9-igličnih i 24-igličnih štampača svih klasa, preko termalnog notebook PN48 ink jet-a, do vrhunskog profesionalnog PROLASER 12. Ono što firmu MERKOM razlikuje od sličnih je razvijena prodajno servisna mreža u koji su uključeni i takvi dileri kao što su „Šumadija", „Jugodata", „Jugolaboratorija", CET, „Tehnika", „Byte" i drugi.

Kontakt adresa: MERKOM, Kruševac, Dom sindikata V/5, tel. 037/28 366

Priprema: Raška Jovanović

Priprema i uređuje: „MICROSYS"

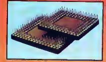
COMPAQ BILTEN

5

INTEGRISANA ZAŠTITA NA RAČUNARIMA COMPAQ DESKPRO

Uočavajući povećanu ulogu i značaj PC računara u složenijim sistemima (tzv. "connected environments"), COMPAQ je među prvim proizvođačima PC računara posebnu pažnju posvetio višestepenoj zaštiti od neovlašćenog korišćenja. Ovde skrećemo pažnju na zaštite primenjene kod dve familije računara: DESKPRO/M i DESKPRO/i.

Postoje tri nivoa zaštite: softverska, hardverska i mehanička. Imajući u vidu ovako širok repertoar zaštita, koje omogućavaju pojedinačno osiguranje pojedinih delova računara, administrator informacionog sistema ili pojedinačni korisnik ima na raspolaganju velike mogućnosti u obezbeđivanju efikasne zaštite sistema. Postoje sledeći mehanizmi zaštite:



Fizičko zaključavanje

Za razliku od onih smešnih univerzalnih bravica sa ključem, kojima navodno možete zaključati tastaturu kod većine delikostnocnih klonova (ali ne i kućište), COMPAQ DESKPRO modeli imaju propisnu bravu na zadnjem panelu, kojom se osigurava kućište od neovlašćenog otvaranja. Ovakva opcija je u našim uslovima veoma zahvalna. Kada stignete na posao, ne morate proveravati da li u računaru imate isti hard disk (odnosno da li ga uopšte imate).

Šifra po uključenju sistema

Logika ove zaštite je dobro poznata: po uključivanju računara potrebno je na upit odgovoriti pravom šifrom. Za razliku od AMI, AWARD i drugih tipova BIOS-a, gde ovu zaštitu možete probiti za par sekundi, kod računara COMPAQ DESKPRO to ne vredi ni pokušavati. Naime, u ovom slučaju inicijalni Setup program za računar je na disketi, koju poseduje samo korisnik.

Šifra tastature

Ova opcija omogućuje korisniku da spreči pristup računaru preko tastature svima koji ne znaju odgovarajuću šifru.

QuickLock/QuickBlank

Kao i prethodna, tako i ova opcija omogućava korisniku da spreči pristup računaru preko tastature svima koji ne znaju odgovarajuću šifru, a da pritom ne napušta aplikaciju u kojoj trenutno radi, što je zgodno za kratke prekide u radu. Ujedno, moguće je i privremeno brisanje sadržaja ekrana (QuickBlank) sve dok je tastatura blokirana. Na ovaj način zaštićeni su i rezultat i tajnost rada od bilo kakvog neovlašćenog pristupa.

Korisne adrese:

MICROSYS
Novi Sad
Tel. 021/611-366, Fax. 021/20-544

MICROSYS, Poslovnica Beograd
Molerova 70
Tel. 011/432-690, 430-059, Fax. 011/432-690

KRATKE VESTI

COMPAQ laserski štampači

COMPAQ je predstavio dva laserska štampača, čime je označio svoj ulazak i u oblast profesionalnih periferijskih uređaja. PAGEMARQ 20 i PAGEMARQ 15 laserski štampači međusobno se razlikuju po brzini štampanja: 20 odnosno 15 stranica u minuti, respektivno. Predstavljaju evidentan COMPAQ-ov doprinos povećanju efikasnosti i udobnosti rada u LAN okruženju. Pored velike brzine štampanja, ovi štampači se mogu priključiti direktno na lokalnu računarsku mrežu koristeći posebnu interfejsnu karticu (NIC - Network Interface Card). Da stvar bude zanimljivija, mogu se priključiti istovremeno na 3 različite mreže: Ethernet, Token ring i AppleTalk. Naravno, postoje i standardni Centronics i RS232C priključci. Svi pomenuti priključci se mogu koristiti simultano.

Nabrojimo još nekoliko karakterističnih podataka: automatska detekcija načina štampanja (Adobe PostScript Level 2 i PCL 5), nekoliko različitih rezolucija (800×400, 400×400, poboljšana 300×300), formati papira A3, A4, A5 i B4; uz svaku NIC isporučuje se upravljački program koji omogućava konfigurisanje, testiranje, upravljanje štampačem, kao i rad sa fontovima koristeći PC sa DOS ili Windows okruženjem ili bilo koju radnu stanicu u mreži. Postoji 50 internih fontova: 35 PostScript, 8 PCL fontova koji se mogu skalirati na željenu veličinu i 7 fiksnih PCL fontova. Za dalje punjenje štampača fontovima na raspolaganju je Programibilni Font Modul, kapaciteta 1 odnosno 2 MB. Kad se PFM jednom napuni, njegov sadržaj je dostupan svim stanicama u mreži, čime se znatno ubrzava rad. Za još veće zahteve na raspolaganju je tvrdi disk kapaciteta 60 MB za smeštanje PostScript fontova, formulara, grafičkih materijala i slično.

Motorna snaga ovih štampača je AMD 29000 RISC procesor, koji radi na učestanosti 20, odnosno 16 MHz, dok je operativna memorija standardno 4 MB, proširivo do 20 odnosno 18 MB. Toner, razvijač i valjak se nalaze u jednoj celini. Jedno punjenje omogućava štampanje 12.000 stranica.

Novi monitor

COMPAQ je predstavio novi monitor visoke rezolucije, vrlo prihvatljive cene. Novi COMPAQ 1024 Color Monitor je idealan za današnje visoko zahtevne Windows aplikacije. Podržava rezolucije do 1024×768 tačaka pri učestanosti osvežavanja 72Hz, bez prepletanja. Proizvodi se sa ekranom dijagonale 14 inča. Standardno se isporučuje sa 9 različitih fabrički postavljenih („preset"- ovanih) režima. Predstavlja idealno rešenje za svaki COMPAQ računar.

SVE COMPAQ-ove 486-ice

Pre no što krenemo sa predstavljanjem najznačajnijih COMPAQ računara, familije SYSTEMPRO napravićemo kratku rekapitulaciju svih COMPAQ računara sa sa familijom procesora 486

Prateći INTEL-ovu orijentaciju diversifikacije procesora 486 COMPAQ je u međuvremenu predstavio nekoliko proizvoda na bazi novog člana ove familije 486DX2/66. Reč je o procesoru kod koga se svim internim operacijama („na cipu") upravlja taktom učestanosti 66 MHz dok se za eksterne aktivnosti („van cipa") koristi učestanost od 33 MHz.

Za svaku vrstu procesora postoji 2 do 4 osnovna modela pri čemu model 1 znači da se radi o konfiguraciji bez diska.

Familija računara Portabl

Familiju COMPAQ portabl računara predstavljaju dva modela sa monohromatskim i dva sa kolor monitorom. Računari su sa EISA magistralom i dva slobodne utičnice. Monitor je dijagonale 10.4" klase VGA, tastatura ima 102 tipke a masa kompletnog računara je 7.9 kg. Namenjeni su pre svega korisnicima koji zahtevaju vrhunske performanse i mogućnost proširenja.

Standardne konfiguracije familije **Portabl** sa procesorom 486 date su u sledećoj tabeli

Model		procesor	RAM (std/max) MB	disk (MB)
486	Model 120 210	486DX/33	4/32	120 210
486c	Model 120 210	486DX/33	4/32	120 210

Familija ProLinea

Zasniva se na ISA magistrali (vidi *Bilten 3/92*). Procesor je sa 8kB interne keš memorije. Standardno je ugrađen grafički kontroler za rezoluciju 1024x768 koji omogućava prikazivanje 256 boja. Kompatibilan sa VGA pri rezolucijama 640x480 i 800x600. Predstavlja pravi „low-price" proizvod po ceni već sasvim konkurentan ponudi „generičkih" kompatibilnih računara, po performansama i kvalitetu na samom vrhu ponude. U osnovnoj nameni predstavlja radnu stanicu odnosno PC računar univerzalne namene i visokih performansi.

Kapacitet diska je moguće povećati do 210 MB. Može se priključiti jedinica trake kapaciteta 120 MB.

Standardne konfiguracije familije **ProLinea** sa procesorom 486 date su u sledećoj tabeli

Model		procesor	RAM (std/max) MB	disk (MB)
4/33	Model 1 84 120	486DX/33	4/16	0 84 120
4/50	Model 1 120 240	486DX2/50	4/16	0 120 240

Familija DESKPRO/i

Zasniva se na ISA magistrali (vidi *Bilten 3/92*). Procesor je sa 8 kB internog keša. Podržava sekundarni keš kapaciteta 64 kB. Standardno je ugrađen integrisani QVision grafički kontroler za rezoluciju 1024x768, koji omogućava prikazivanje 256 boja. Ugrađen je Poslovni Audio sistem („Business Audio System - BAS"). Personalni računar u osnovi opšte namene zasnovan na ISA arhitekturi. Predstavlja povoljan izbor za sledeće klase primena: grafička radna stanica za niže zahteve, mrežni server za male mreže, personalni računar za visoke zahteve.

Kapacitet diska je moguće povećati do 510 MB. Može se priključiti jedinica trake kapaciteta do 320/525 MB.

Standardne konfiguracije familije DESKPRO/i date su u sledećoj tabeli

Model		procesor	RAM (std/max) MB	disk (MB)
4/25s	Model 1 84 120	486SX/25	4/32	0 84 120
4/33i	Model 1 120 240	486DX/33	4/32	0 120 240
4/66i	Model 1 120 240	486DX2/66	4/32	0 120 240

Familija DESKPRO/M

Personalni računar u osnovi opšte namene baziran na EISA [illegible] i potpunoj modularnosti (vidi COMPAQ Bilten 1/92). U našim uslovima vezano za razloge koji su već apsolvirani kod prethodnih familija modela predstavlja povoljan izbor za sledeće primene: grafička radna stanica za visoke zahteve, mrežni server za male i srednje mreže, personalni računar za visoke zahteve.

Standardne konfiguracije familije DESKPRO/M date su u sledećoj tabeli

Model		procesor	RAM (std/max) MB	disk (MB)
486s/25M	Model 1 60 170	486SX/25	4/64	0 60 170
486/33M	Model 1 120 340	486/33	4/64	0 120 340
50M	Model 1 120 340	486DX2/50	4/64	0 120 340
66M	Model 1 120 340	486DX2/66	4/64	0 120 340



Familija SYSTEMPRO/LT

U nomenklaturi COMPAQ-a ova familija se definiše kao specijalizovani mrežni server. Međutim u našim trenutnim tržišnim uslovima definicija se koriguje u SYSTEMPRO za manje zahteve sa svim aspektima izvorne familije osim multiprocesiranja. Razlozi za ovakav pristup su višestruki: povoljniji odnos performanse/cena, prosečno veoma niski zahtevi koje korisnici postavljaju za „mainframe" računare, relativno uzak krug korisnika koji se odlučuju za višeprocesorsku varijantu.

Standardne konfiguracije familije SYSTEMPRO/LT date su u sledećoj tabeli

Model		procesor	RAM (std/max) MB	disk (MB)
486SX/25	Mod 210 510 680	486SX/25	8/64	210 510 680
486/33	Mod 210 510 1020	486/33	8/64	210 510 680
486DX2/50	Mod 210 510 1020	486DX2/50	8/64	210 510 1020
486DX2/66	Mod 210 510 1020	486DX2/66	8/64	210 510 1020

Familija COMPAQ SYSTEMPRO

Predstavlja vrhunski mrežni server i višekorisnički računar u klasi srednjih sistema kao idealnu zamenu za već zastarele i poddimenzionisane „mini" računare.

Procesor ima 8 kB interne i 512 kB sekundarne keš memorije. Svi modeli su opremljeni IDA-2 disk kontrolerom pri čemu kod modela sa sufiksom „c" („enhanced") disk kontroler ima keš kapaciteta 4 MB. Postoje velike mogućnosti proširivanja svih računarskih resursa od kojih moramo posebno da pomenemo varijantu dvoprocesorskih sistem.

Model		procesor	RAM (std/max) MB	disk (MB/broj diskova)
486	Mod 240	486/33	8/256	240/2
486	Mod 420	486/33	8/256	420/2
486	Mod 840	486/33	8/256	840/4
486	Mod 2040	486/33	8/256	2040/4
486	Mod 1c	486/33	8/256	0
486	Mod 680c	486/33	8/256	680/2
486	Mod 1020c	486/33	8/256	1020/4
486	Mod 2040c	486/33	8/256	2040/4

DESET DANA NA TAJVANU

U Tajpeju, glavnom gradu Tajvana, ponudu možete dobiti i pre jutarnjih novina, a možda i pre petlova. U Tajpeju ne možete dobiti besplatno uzorak nekog proizvoda, hrana je odlična, domaćini ljubazni, vlaga velika, a za one koji vole da se cenjkaju to je obećana zemlja, posebno zbog „ugrađenih" pogodnosti. Evo kako.

Dejan V. Veselinović

Stići do Tajpeja uopšte nije lako. Nekada se iz Beograda letelo za Bangkok ili Singapur, odakle se mogao uhvatiti vezni let za Tajpej (retki su direktni letovi od Tajpeja). Danas, međutim, zbog blokade treba drumom ići do Bukurešta ili Sofije, pa odande uhvatiti avion za Bangkok ili Singapur. Tu treba ostati odvoljno dugo da se dobije tajvanska viza, pa tek onda stići do odredišta.

Tajpej je moderan grad sa oko 4 miliona stanovnika, po arhitekturi očigledno građen po uzoru na američke gradove, sa velikim i veoma dugačkim bulevarima koji se međusobno presecaju pod pravim uglovima. Sve je dosta novo i moderno, što nije čudno ako se zna da je izgradnja otpočela tek pre nekih 40 godina. Grad diše baš kao i drugi veliki gradovi današnjice. Saobraćaj je izuzetno živ, a o komercijalnom životu da i ne govorimo.

Naravno, za nas je Tajpej najvažniji kao carstvo industrije računara i računarske opreme, sa izuzetkom tvrdih diskova i štampača. Činjenica je da je najveći broj računarskih firmi lociran u ili oko samog grada, a posebno u nekoliko razvojnih parkova u neposrednoj okolini grada. Uprkos tome, za devet dana odravka videli smo slovom i brojem tri radnje sa računarskom opremom. U bankama, na primer, standardna oprema je Acer računar i Fujitsu štampač, ali po definiciji uvek u ugašenom stanju, dok službenici i kurirn trčkaraju unaokolo sa tonama papira.

Problem je, kako smo saznali, u totalno drugačijoj koncepciji rada na kineskom. Naime, po kineskom pismu, piše se odozgo nadole, a isto to primenjeno u računarima ide sa leva na desno (znakovi nisu poseban problem, ali podrazumevaju osetno poodjšanu grafiku, čak i u slovnom rezimu), to je odvoljno da za većinu ljudi upotreba računara, uz ostale svuda uobičajene probleme, postane nepremostiva prepreka. Čudno, ali istinito, da je najveći svetski centar za proizvodnju računara tako slabo opremljen sopstvenom robom.

TATA, MAMA, SIN

Najčuvanija tajna na Tajvanu je stvarni izvor opreme. Naime, veoma mali broj prodavaca je ujedno i proizvođač – oni jednostavno kupuju proizvode od fabrika koje opet ne žele da se bave prodajom i trgovinom već da robu što pre pretvore u novac. Tako se često dešava da se jedan te isti proizvod nađe u više kataloga pod različitim oznakama. Naravno, čak i letimičan pregled će lako pokazati zajedničko poreklo.

Nije po sredi samo puki komoditet proizvođača i ljubav za trgovinu, koja objektivno postoji, već čista praktičnost, u slučaju većih proizvođača, recimo monitora, mesečne proizvodne kvote iznose i po 50 000 primeraka. Da bi sve to prodali morali bi držati velika i skupa odeljenja za prodaju, ali bi uz monitore trebalo prodavati i sva ostalo, što bi opet bio problem za proizvođača. Dakle, radi se i o izvesnoj podeli rada, normalnoj pojavi na razvijenom tržištu, koja ne samo da je korisna nego čak i nužna.

sve promenljive na početku funkcije, sada ih možemo deklansatr baš tamo gde nam trebaju. Recimo

Međutim, broj tako velikih proizvođača bilo kog dela opreme je mali, dok je broj malih ili usko specijalizovanih proizvođača daleko veći. Oni ostvaruju visoki kvalitet, neretko u strogo porodičnom aranžmanu, gde je otac direktor, mama šef prodaje a sin – inženjer – razvija proizvode, plus još pokoji zaposleni.

Primer takve firme je cenjeni proizvođač isključivo komunikacione opreme, firma Logidata. Ovo je porodična firma, u kojoj tata i sinovi rade sa drugim zaposlenim, a mama ipak ostaje kod kuće. To im nije ni najmanje smetalo da sa svojim proizvodima izbiju na sam vrh kvalitete proizvodnje (stvarne proizvodnje – prave ih sami) modema i faks/modem kartica. U svim testovima na koje smo naišli u svetskoj štampi, proizvodi ove firme su uvek bili pri samom vrhu, ako već ne i prvi. Jednostavno, stav je da sve, a ne samo poneki udarni model, mora biti na istom nivou kvaliteta. S druge strane, pošto se radi o „relativno" maloj firmi, broj proizvoda je ograničen kako po upotrebi tako i po apsolutnom broju, radi se faktički o četiri grupe proizvoda koje se raznim varijacijama šire na paletu od po 5-6 modela unutar svake grupe.

Primer druge vrste proizvođača bi bila i kod nas poznata firma Longshine, koja je za tajvanske pojmove čak dosta velika, iako se ne može porediti sa zaista velikim igračima kao što su Acer i Mitac, čiji se kapital već meri milijardama dolara. Longshine spada u red velikih za Tajvan i jačih srednjih firmi u svetskim razmerama. Centrala na Tajvanu godišnje obrne oko 45 miliona dolara, a sa ograncima u nekoliko zemalja u svetu i celih 100 miliona dolara.

Taj efekat „relativne" veličine se najodije ogleda u broju inženjera koji rade u razvoju, a koji prelazi brojku od 70 ljudi. Imali smo prilike da vidimo radne laboratorije tih ljudi kao i njihove radne uslove: tamo se zaista veoma ozbiljno radi i teško je postaviti pitanje koje ostaje bez odgovora.

KO ŠTA RADI

Broj razvojnih projekata ujedno ukazuje i na najnovije trendove. Recimo, trenutno se izbacuju poslednji primerci starije generacije monitora, čije mesto zauzimaju novi modeli sa kvalitetnijim elektronskim cevima i, iznad svega, sa sve odlijom zaštitom od zračenja. Interesantno je da se ta zaštita dosad uglavnom odnosila na deo ekrana okrenut ka korisniku, ali na i na druge strane, sada se, međutim, prelazi na sveobuhvatnu zaštitu sa svih strana, dakle ne samo zaštitu korisnika već i okoline.

Najnovije generacije IDE kontrolera se sve jasnije razgraničavaju na dve grupe, standardne i inteligentne. Prva grupa sve češće obuhvata i kontrolere disketnih jedinica koji rade i sa najnovijom verzijom disketa od 3.5 inča kapaciteta od 2.88 MB, dok se druga grupa deli na ISA kontrolere samo za IDE diskove, ISA kontrolere za SCSI diskove, mešane (IDE i SCSI) kontrolere za ISA sabirnicu i, naravno, verzija za EISA sabirnicu. U slučaju ISA sabirnice, brzine transfera se kreću oko 4 MB/s za SCSI i 5 MB/s za IDE diskove sa brzinama sabirnice do 12 MHz, dok za EISA sabirnicu brzina raste i do 16.5 MB/s. Ako je prvo veoma, veoma brzo, drugo je zaista trenutni odziv kojim se stiče utisak da se radi o RAM disku a ne o mehaničkoj jedinici: to zaista treba videti – čoveku zastane dah.

Poslednji krik tehnike je, naravno, lokalna sabirnica (*local bus*) kada su dodatne kartice neposredno povezane za procesor na njegovom radnom taktu, a ne posredstvom standardne ISA ili novije EISA sabirnice. Sada se pojavljuju proizvodi sa dve utičnice za video karticu, pri čemu je druga utičnica namenjena specifičnim IDE disk-kontrolerima. Prema najavnoj literaturi, brzine prenosa sa i na IDE disk rastu i od 4 puta, što je za najsporiji deo inače veoma bitne komunikacije unutar računara veoma važno.

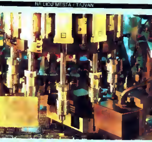
Što se video kartice tiče, postalo je jasno da se ona maltene linearno ubrzava sa povećanjem radnog takta, samim tim, ako se video čip stavi na karticu koja izdržava velike radne taktove od recimo 33 MHz, to je ravno 4 puta brže nego na standardnoj ISA sabirnici, da i ne govorimo o ukinutim stanjima čekanja, potrebnim ciklusima i slično. Za sada, dva najpopularnija video čipa na ovakvim karticama su proizvodi firmi Tseng (podverzija poznatog video čipa ET4000) i NCR (77C22E+). Primera radi, brzine prenosa se kreću oko 25 MB/s. Sa ovakvim karticama, čak ni specijalizovani proizvod, kao video čip firme S3 sa oznakom 86C911, nema šta da traži, jer zaostaje po brzini za faktor veći od 2:1.

Iako postoji niz reklama i još više šarenog papira na ovu temu, gotovih proizvoda je malo. Naime, sasvim je uobičajena praksa na Tajvanu da se o tek odlazećim proizvodima priča kao o gotovoj stvari, iako se radi o „živim" primercima po laboratorijama i na sajmovima, dakle o uzorcima koje možda i možete isprobati, sve te stvari se još uvek ne proizvode masovno, odnosno tek ulaze u iole masovniju proizvodnju.

Problemi se odnose na standarde (te priče smo se bar naslušali!) jer svako hoće da ima svoj i samo svoj standard, kao i na probleme oko samog procesa proizvodnje. Oni nisu tehnički nerešivi (mada i tu ima specifičnih teškoća) već se tiču veličina serija koje su tamošnjim proizvođačima potrebne da bi opstali: naime, ona moraju biti takve da se proizvodnja brzo isplati, a opet ne mogu postati velike sve dok se proizvodnja na uhoda. Dakle, stalno se vrtimo oko problema kokoške i jajeta, kao i mnogo puta do sada.

DISKOVI I PORTABLI

Najuže grlo tajvanske računarske industrije su tvrdi diskovi, koje oni tek počinju da proizvo-



da lokalno. Za sada mahom se kupuju iz Singapura u koma se proizvodi oko 87% ukupne svetske proizvodnje tvrdih diskova. Naime, svako ko nešto znači u toj oblasti ima fabriku u Singapuru, od Seagate-a do Miniscribe-a (izuzetak su japanski proizvođači i firma Quantum). Zato se za sada diskovi kupuju u Singapuru, a lokalna industrija se udružila na samom Tajvanu i uložila dosta novaca u razvoj domaće proizvodnje. Naravno, neće im biti lako, jer tek treba da ubede svet da poznaju i umeju da proizvedu mehaniku na istom nivou kvaliteta kao i suvu elektroniku. S druge strane, i namaju naki izodr ako žele da zaokruže sopstvenu proizvodnju.

Portabli u A4 formatu već se proizvode lokalno. Ponuda postoji, a stvarna konkurencija je samo Japan, od koga su, opet, vezani da kupuju LCD prikaze. Njih prave firme kao što su Epson, Sanyo, Hitachi, NEC, Toshiba, Sharp i Citizen, pri čemu je po nama najlošiji onaj prvi (malo čudno od tako poznate firme) a najbolji model firme Sharp. Videli smo i prikaze u boji. I mislimo da se oko njih podigla velika prašina, daleko veća nego što realno zaslužuju, boje još uvek daleko zaostaju za klasičnim monitorima, mada su ti prikazi, zbog upotrebe aktivnih umesto pasivnih matrica, objektivno daleko brži prilikom osvežavanja, na oko gotovo isto onako brzi kao i monitori.

Što se srca mašina tiče, modeli sa procesorom 80286 su definitivno nestali – potpuno su ih zamenili modeli 80386SX. Radni taktovi od 16 MHz su već retkost, 20 MHz je standard, a uskoro će ga zameniti i celih 25 MHz. Međutim, to nije sve – nudi se sve veći broj istovetnih mašina sa procesorima 80486SX na 25 MHz, pa bogami i procesorom 80486DX na 33 MHz, na žalost, cene su još uvek veoma visoke, ali treba očekivati njihov skori pad u realnije vode (bar za druge, a kako će biti nama, živi bili pa videli).

U oblasti prenosnika, najljući konkurenti su još uvek Japanci, koji se baram na tom polju na predaju, tamo postoji niz firmi za koje niko nikada nije ni čuo. Specijalizovale su se za proizvodnju, dok prodaju prepuštaju drugima. Dakle, oni prave i po potrebi lepe bilo čije nalepnice. Koliko smo imali prilike da vidimo, japanska roba je još uvek bolja od tajvanske. Koliko dugo će tako i ostati, videćemo, jer su se Tajvanci opasno naoštrili za konkurenciju u tom polju. Cene već sada padaju, a sve prognoze pokazuju da će padati i dalje.

LICE I NALIČJE

Iza cele ove zaista impresivne priče uvek postoji i naličje, o koma se po pravilu daleko manje zna nego o licu. U slučaju Tajvana, naličje se sastoji od blago rečeno žestoke konkurencije i inovacije.

Već sada, konkurencija je oterala profitne stope na opasno niske nivoe: recimo, jedan proizvođač video kartica mesečno proizvede 50-60 000 komada sa čistom zaradom od svega jedan dolar po komadu, to je zarada manja od 2% vrednosti najjeftinije kartice, što je poslovanje na ivici bržaća. Izodra nema, jer je konkurencija veoma oštra. Nije mnogo odlje ni sa proizvođačima matičnih ploča, a još ponajodlje stoje proizvođači metalnih kućišta, kojima je najveća pretnja najnovija ponuda potpuno plastičnih kućišta, koja teže da zauzmu niže segmente tržišta. Niko ne sumnja da će do toga i doći za godinu dve dana.

Po nekim ispitivanjima, oko 40% rukovodilaca firmi smatra da je ključni problem tajvanske računarske industrije inovacija. Naime, smatra se da su Tajvanci sjajni za masovnu i jeftinu proizvodnju, ali da oni samo preuzimaju od drugih ideje koje potom komercijalizuju. To je dobro na kraći, ali ne valja na duži rok. I baš tu se nalaze najveće klopke razvoja, jer inovacija podrazumeva velika ulaganja u istraživanja, a da bi to postalo moguće, profitne stope moraju biti veće no što su sada. I tako se opet vraćamo na problem kokoške i jajeta, ali se u ovom slučaju radi o opstanku. Pre dve godine manje od 10% rukovodilaca je ovaj problem smatralo ključnim, a danas ih je četiri puta više. Nema sumnje da će s vremenom njihov broj rasti jer je konkurencija jaka a svetsko tržište nije beskrajno veliko. Najzad, oni čija su potrebe najveće, nerazvijene zemlje, muku muče kako odnosno od čega to da plate, pa zato ostaju samo potencijalna tržišta. Sa druge strane, u SAD se traži visok nivo tehnologije pa standardna oprema sve teže prolazi, dakle, najveće tržište insistira na sve većem stepenu inovacije. Kako će to da reše, videćemo.

Sve u svemu, Tajvan kao svetska sila u industriji računarske opreme nema namera da odstupa, već naprotiv, da se učvišćuje pa čak i širi. Kuda? Pa na istok, naravno, a tamo se po njihovom viđenju stvari nalazimo i mi.

SAJMOVI

Novi Sad: VRT'92

Sajam vrhunske računarske tehnologije



Snimio: Dragan Žugić

Pod ovim izazovnim imenom krije se namera grupe profesionalaca iz novosadske agencije TEGA, potpomognute sa nekoliko firmi pokrovitelja i još većeg broja firmi učesnika, da se u SR Jugoslaviji ustanovi manifestacija na kojoj će se okupljati domaći proizvođači i isporučioci mašinske i programske opreme iz oblasti informacionih tehnologija, kao i profesionalci iz industrije i akademskih institucija.

Pored klasičnih sadržaja (izložba, promocije novih proizvoda i usluga), planirano je da se održavaju stručni seminari iz širih aktuelnih oblasti ili pak iz specijalizovanih užih stručnih domena. Ovo je suština dogovora postignutog na okruglom stolu koji je organizovan na početku manifestacije na inicijativu računskog centra INFORMATIKA iz Novog Sada uz učešće predstavnika većeg broja firmi (Elektrovojvodina, NIS PTT, Microsys, Pokrajinsko Izvršno Veće, Sekretarijat za nauku i informatiku, Info-D, da pomenemo samo neke učesnike) u organizaciji i koordinaciji Tehnološke galerije TEGA.

Od iduće godine očekujemo da se u ovo vreme organizuje INTER INFO umesto dosadašnjeg VRT-a.

Ova manifestacija (Novi Sad, 07. – 10.10.1992.) se održava već četvrti put, a drugi put pod ovim imenom. Pomenimo i pokrovitelje: Generalni sponzor – MICROSYS iz Novog sada, zatim Zlatni pokrovitelj – INFO D iz Beograda, Pokrovitelj za oblast softvera – PLUS SISTEMI, Pokrovitelj za oblast grafike – OSA INŽENJERING iz Beograda i Pokrovitelj za oblast ličnih računara – MOST iz Beograda.

Sajam je po tradiciji prva dva dana bio zatvorenog tipa, za posetioce sa pozivnicama, dok je u narednim danima bio otvoren za sve zainteresovane. U toku drugog i trećeg dana održana su predavanja sa diskusijama prema sledećem rasporedu:

- Višekorisnički sistemi na bazi personalnih računara (S. Malić)
- Poslovne aplikacije – izazovi i rešenja – PROGRESS RDBMS/4GL & TOOLS (V. Orović)
- Uvod u bazu podataka MAGIC II (M. Kuknka)
- Uvod u geografske informacione sisteme (GIS) ARC/INFO i ARC/VIEW (B. Gavrić i D. Đikić)
- CASE alati u projektovanju informacionih sistema (Đ. Janković)
- Geografski informacioni sistemi (D. Žugić)
- Napredne tehnike u AutoCAD-u (M. Micić Mimi)
- EDM – Sistemi za prenos tehničke dokumentacije u računar i njeno arhiviranje (P. Damjanović i B. Damjanović)

U izložbenom delu sajma izlagači su predstavili svoje proizvodne programe i usluge koje nude potencijalnim korisnicima. Promovisani su novi modeli, novi softverski paketi, konsalting i inženjering usluge. Posebno izdvajamo promociju COMPAQ računara (modeli DESKPRO/M, ProLinea, SYSTEMPRO/LT) na standu MICROSYS-a, PROGRESS RDBMS/4GL & TOOLS u organizaciji firme ABV Information Systems na štandu firme MICROSYS, opremu za inženjersko projektovanje i arhiviranje tehničke dokumentacije firme OSA INŽENJERING, štampač EPSON LQ 100 na štandu firme Info-D itd.

Sajmu je bila posvećena odgovarajuća medijska pažnja angažovanjem RTV Novi Sad i lokalne radio stanice 202, a Sezam je kroz konferenciju VRT92 otvorio stalni kanal za obaveštavanje javnosti o zbivanjima na ovoj skromnoj ali zanimljivoj manifestaciji.

(S. Malić)

SVE MANJI, SVE BOLJI

Na početku su postojali pravozivi računari, preveliki i preteški za svakodnevno nosanje unaokolo, ali ipak prenosivi; njih su zamenlie prenoaiva mašina, oaetno manjih dimenzija i težine, naravno aa rastućim mogućnoatima, a ova opet još manje mašine, formata papira A4. Pogiedajmo dokle su atigii ovi poalednji.

Dejan V. Veselinović

Postoji jedan segment korisnika i ne baš mali niz potraba koji podrazumevaju stalno prisustvo računara ali bar mogu u mnogome dobiti tim stalnim prisustvom Industrija se odazvala pozivu (ujedno i izazovu) i proizvela računare koji se mogu prenositi makar i sa mukom Naravno tehnologija napreduje pa su i ovakve mašine postajale sve manje a sve moćnije uz uobičajeno ponašanje krive cena – na početku veoma visoko a potom sve niže i niže

Problem sa ovom oblasti računarstva je oduvek bio u odnosu cene i relativne moći ponuđene platforme Donedavno cena se kretala osetno iznad relativne snage mašine to je onemogućavalo ovakvim proizvodima da postanu ne druga ili pomoćna već jedina korisnikova mašina Isto tako svi su obećavali dok je malo ko zaista i nudio mogućnosti proširenja kako unutar same mašine tako i u vidu dopunske stanice kod kuće ili na poslu koja bi mogla primiti razne kartice i poneku hardversku jedinicu (recimo drugi tvrdi disk)

Danas su ti problemi u velikoj meri prevaziđeni Nije više neko čudo imati prenosnik sa procesorom 80486SX pa i DX tvrdim diskom od 120 MB pa čak i impresivnim prikazom Međutim ovo je domen mahom veoma velikih firmi kao što su Toshiba, NEC i Sharp a cene odmah eliminiše iz igre većinu kupaca Njima ostaje da se zainteresuju za računarski slabije proizvode ali sa osetno manjom cenom

Prilikom nedavnog boravka na Tajvanu imali smo prilike da vidimo prilično veliki broj ovakvih mašina iz raznih izvora uprkos naizgled velikoj sličnosti između ovih proizvoda postoji i dodar stepen različitosti od malih i malo bitnih do popriličnih a pre svega u ceni Naš stav je oduvek bio da kvalitet (kao sveobuhvatni izraz osobina mašine) treba posmatrati u kontekstu cene Naime za male računare se uvek mora platiti od 50 do 100% više nego za ekvivalentnu stonu verziju slične mašine – to je cena pranosivosti i malih dimenzija No to je tek početak razlika ispod haube ima napretek.

OPIS

Longshine N300 A4 pranosnik zaista zauzima prostor jednog lista papira formata A4 štaviše računar je malo širi ali i malo kraći od lista papira Mat crne boje ima veoma ravne ivice i deluje gotovo aerodinamično (dođe nam da mu izmerimo koeficijent otpora vazduhu Cx) Disketna jedinica (3.5 inča 1.44 MB) se nalazi sa prednje strane u desnom donjem uglu a levo od nje je otvor za priključenje spoljne tastature Na levom boku računara nalaze se vratanca za akumulator prekidač za uključenje i isključenje i mali koaksijalni priključak za spoljno napajanje i/ili punjenje akumulatora Na desnom boku se nalaze jedne vratanca koja štite RS-232C priključnicu a na poleđini se nalaze priključnice za spoljnu kutiju sa proširenjima paralelne veza koja se može pretvoriti i u vezu za spoljnu disk jedinicu od 5.25 inča i 1.2 MB kao i priključnica za spoljni video prikaz

Odmah ispod brave na prednjoj strani (koja je uzgred neobično široka – valjda zato da bi efektno prekrila dopunski otvor) nalazi se mesto za proširenje memorije U ovom slučaju memorija se proširuje pomoću posebnih kartica nalik kreditnim (istog formata ali debije) Sve je suviše malo za standardne SIM module pa se pribeglo ovakvom rešenju radi se o proizvodu koji je sve popularniji u industriji iako se oslanja na veoma labave standarde Međutim usled zaista malog prostora koji traži kao i usled tehnologije koju podrazumeva ovo je de fakto standard prema kome se već radi a ubuduće će se sve više raditi Ovaj model se serijski isporučuje sa 2 MB a jedino moguće proširenje je do 4 MB za većinu korisnika sasvim dosta.

Ovaj pristup ima svojih vrlina i mana. Vrlina su mali prostor i lakoća instalacije bez potrebe da se otvara sam računar dok je mana nenadogradivost (tj ako vam treba više bacate staru i kupujete novu karticu) Što se cene tiče ona je naravno osetno veća nego kod standardnih oblika memorijskih proširenja i to za faktor od solidnih 2.2:1 To je cena prenosivosti

Uz računar ćete dobiti i veoma lepo i čvrsto skrojenu kesu za prenošenje u kojoj ćete naći jedinicu za spoljno napajanje i/ili punjenje akumulatora (15 V 2 A, naravno univerzalnog tipa 110-240 V sa automatskim prilagođavanjem) kabl za povezivanje na mrežu kabl za prebacivanje sa DB25 na DB9 za serijsku vezu priručnik i jednu disketu sa raznim korisničkim programima

TEHNOLOGIJA

Računar za osnov koristi Intel procesor 80386SX a radni takt je 20 MHz Pošto nema keš memorije ostaje vam da se dovijete sa standardnom memorijom Serijski računar ima 2 MB organizovana u dva reda od po 1 MB čime se omogućava rad sa celim stanjem čekanja Ako imate dopunsku memoriju (kao u ovom slučaju) tih dodatnih 2 MB omogućava rad sa dvostranim prepitanjem čime se broj stanja čekanja smanjuje sa 1 na 0.4-0.7 uz poboljšanje brzine rada



Slika 1

Hardverske i softverske performense



Slika 2

Brzine prenosa za tvrdi disk

Što se radnog takta tiče on i jeste i nije dovoljan Činjenica je da se sve više koristi nešto veći radni takt od 25 MHz ali se za sada to naplaćuje posebno sa druge strane još se uveliko prodaju modeli sa radnim taktom od 16 MHz sasvim normalno uz odgovarajuće popuste Ovaj model se dakle nalazi u "zlatnoj sredini" a takva mu je i cena Smatramo da je za većinu korisnika ovakvih mašina 20 MHz sasvim dovoljno a treba imati u vidu i to da veći takt brže prazni akumulatore drugim rečima ako ćete na ovakvoj mašini mahom kucati tekst najbolje bi bilo da mu oborite radni takt na što manju učestanost jer tima dobijate na trajanju akumulatora.

Postoji mogućnost ugradnje numeričkog koprocesora ali se to mora uraditi u fabrici – definitivna mana kako god to gledali

Prikaz koristi CCFT (*Cold Cathode Fluorescent Tube*) LCD tehnologiju sa ivičnim osvetljenjem to je najčešći oblik prikaza koji se danas koristi U ovom slučaju prikaz je potpisala firma Sharp što se po kvalitetu i vidi a ne začuđuje mnogo s obzirom da je proizvod po narudžbini napravljen u Japanu Radi se o poslednjoj generaciji pasivnih prikaza u kojima jedan aktivni element (tranzistor) multipleksiranjem kontroliše više (tri) piksela na ekranu ovo je čisto ekonomski kompromis jer bi aktivni prikaz (u kome postoji po jedan tranzistor za svaki piksel) bio osetno brži ali i osetno skuplji pa se ova tehnika uglavnom koristi za prikaze u boji

I tehnologija tvrdih diskova je sasvim nova Radi se o najnovijoj generaciji jedinica sa prečnikom od svega 1.8 inča (u poređenju sa standardnim jedinicama sa prečnikom od 3.5 inča) koje postoje u nizu od 40 60 80 i 120 MB Istina ovaj poslednji se više obećava no što se može stvarno naći ali Bože moj – to je već standardno ponašanje u računarskoj industriji U našem slučaju disk ima formatirani kapacitet od 42 MB

Najzad tastatura Odavno govorimo da su dva elementa računarskih sistema sa kojima je korisnik uvek u dodiru ma šta radio tastatura i prikaz U slučaju malih prenosnika tastatura je najčešće najveći danak apsolutnom kvalitetu jer usled malih dimenzija i trke za što manjom težinom tasture mora biti mala i lagana a to se postiže upotrebom membranskih prekidača Njih nikada nismo voleli iako tehnički gledano nema razloga da ne budu jednako dobri kao i mehanički U ovom slučaju radi se o najboljoj tastaturi membranskog tipa na koju smo do sada naišli mada ima svojih bubica u kontekstu rasporeda tastera Ipak ostavlja utisak veoma solidno urađene tastature što za pranosnik uopšte nije svejedno

PERFORMANSE

N300 je dao rezultate koji zaostaju za apsolutnom referencom (80386SX na 25 MHz Quantum LPS52 disk) nikakvo čudo s obzirom da apsolutna referenca radi na 25% većem taktu Ono što nas je prijatno iznenadilo jeste stepan zaostajanja, koji je i u mašinskim i u programskim testovima manji od razlike u radnom taktu

Prosečan indeks brzine na mašinskim testovima je iznosio 0.79 odnosno 21 % sporije od reference (sa 25% većim taktom) a prosečan indeks brzine na programskim testovima je iznosio takođe 0.79 Ovo poklapanje je veoma siguran dokaz da se radi o dobro projektovanom uređaju koji je pravljen sa razumno ujednačenim delovima i podsistemima i to za realne zadatke a ne za *benchmark* testove

Interesantna su merenja performansi diska po jednom testu njegovo vreme direktnog pristupa iznosi oko 24 mS a po drugom testu 18.05 mS Ovo je prvi put da se susrećemo sa iole većom razlikom u merenju iste stvari a objašnjenje nalazimo u organizaciji lokalnog keširanja na samom disku Najverovatnije je



čan sem u slučaju pomenuta dva kursorska tastera. Srećom proizvođač je "Fn" taster, onaj kojim se menja funkcija ostalih tastera, postavio tamo gde mu je pravo mesto, u krajnji levi donji ugao, a svaka čast na ideji da se ovaj taster malo poveća u odnosu na ostale.

Komande za osvetljenje i kontrast su veoma delotvorne, sa širokim pojasom dejstva, tako da prikaz možete prebaciti sa potpuno zatamnjenog na potpuno osvetljen. Pošto i sam prikaz ima odličan kontrast, kvalitet prikazanog materijala može omesti samo sunčeva svetlost koja pada direktno na prikaz. Kao i kod ostalih prikaza ove vrste, ugao posmatranja je veoma mali, jedva oko +/- 30 stepeni, što je uobičajeno za ovu tehnologiju.

Programska podrška samoj mašini je vanserijski dobra. Recimo, imate mogućnost da celu mašinu bacite u stanje privremenog sna (Shift–Ctrl+S): jednostavno, sve se pogasi, računar i procesor koji padne u polukomatozno stanje, ali zato troši izuzetno malo struje. Time dobijete vremena da sredite svoje misli, a prvi pritisak na bilo koji taster vraća računar u život i na mesto gde ste bili kada ste ga zamrznuli.

Drugi odličan program je onaj kojim se kontroliše mašina kao celina (Ctrl - Shift - M). Njime određujete radni takt (8/12/16/20 MHz), vreme za koje će se prikaz i tvrdi disk ali i sistem kao celina ugasiti usled neaktivnosti, stanje naboja u akumulatoru i slično. Mnogi nude ovakve programe ili u SETUP-u ili kao korisničke, ali ovo je prvi put da vidimo da sve ame baš sve zaista i radi kako treba.

Najzad, dopada nam se i niz sitnica na mašini. Recimo, ideja da se serijska veza izvuče sa desne strane kućišta umesto pozadi se pokazala kao veoma korisna za priključenje laks/modema, posebno u slučaju modela koji se neposredno utiču u sam veznik. U takvim slučajevima, inače poželjnim zbog odsustva kablova, daleko je bolje videti uređaj sa strane nego zavirivati iza prikaza. Slično važi i za veznik za spoljnu tastaturu: on se nalazi sa prednje strane kućišta, odmah ispod njegove tastature, pa je veoma lako priključiti i isključiti spoljnu tastaturu ili miša za PS/2 sisteme. Najzad, po našem ukusu, bolje je da je priključak za spoljno napajanje negde sa strane nego pozadi, a ovde je baš to slučaj: opet, lakše je sve obaviti spreda nego se naginjati iza prikaza.

Priručnik ima 90 strenica sa sadržajem i indeksom, u celini nije loš, sem što su nazivi korisničkih programa koji su u njemu spomenuti izmenjeni.

ZAKLJUČAK

Longshine N300 je veoma dobar model savremen predstavnik svoje generacije prenosnika. Nije najbrži na tržištu, ali jeste od sada najbrži u klasi SX mašina na 20 MHz. Nema krupnijih mana – sve su zamerke uglavnom stvar ličnog ukusa, sa izuzetkom one o poklopcima. Što se vrlina tiče, ima ih dosta, a podvlačimo kvalitet prikaza i tastature: po nama to su veoma važne stvari na svim računarima, a posebno na smanjenim modelima koji nužno nose ograničenja u udobnosti rada. Kada već moraju biti smanjeni, ne daj Bože da budu loše izvedeni, jer onda rad postaje prilično neudoban; u ovom slučaju sve jeste smanjeno, ali rad nije neudoban, čak naprotiv.

Ova mašina bi po funkciji i udobnosti trebalo da zadovolji sve potencijalne korisnike ovakvih uređaja, osim možda one najviše klase korisnika – onih kojima treba više od 4 MB memorije, veća brzina procesora i veća računarska snaga uopšte: takvi će morati da se opredele za neku drugu mašinu, verovatno na bazi procesora 80486 SX ili DX, ali će zato morati i dobro da odreše kesu.

Sa 90-100 minuta autonomije u trajnom radnom režimu, a sa programiranim gašenjem i daleko duže, malom težinom, veoma solidnom izradom i sasvim ubedljivim performansama za svoju klasu, *N300* je veoma pogodan za sve ostale korisnike, uključujući i profesionalce. Čak i u ova nesigurna vremena, projektovana cena bi trebalo da bude niža od trenutno najboljih, pa je to razlog više da se zainteresujete za ovaj model.

da disk ima ugrađeno višesegmentno keširanje sa malo neobičnim algoritmom koji, eto, uspeva da zbuni i testove. Što se opšte organizacije tiče, sam oblik krive prenosa jasno pokazuje da je keš organizovan tako da optimizira čitanje malih datoteka: sve do datoteka od 24 KB brzina prenosa raste, a nakon toga pada otprilike na polovinu vrednosti (jer se keš zasićuje, pa disk radi praktično bez keširanja).

Video podsistem koristi *Cirrus Logic* video čipove koji su se nametnuli kao de fakto standard za prenosnike. Razlog tome je njihova jedinstvena sposobnost da na CCFT prikazima isteraju više nijansi sivog od bilo koje druge kombinacije ili paketa video čipova, a danak toj pogodnosti se očitava u relativnoj sporosti rada. Istina, pravi je doživljaj kada se sa jedne 486-ice pređe na N300, pa se ovaj komentar mora shvatiti kao krajnje relativan i subjektivan.

Sve u svemu, imajući u vidu razliku u rezultatima koja je objektivno manja od razlike u radnom taktu, može se zaključiti da je N300 iznad proseka sopstvene klase.

Pod normalnim, potpuno radnim uslovima (tj. bez programiranih isključivanja) akumulator obezbeđuje dovoljno energije za neprekidnih devedesetak minuta. Međutim, i tu ima raznih mogućnosti: sistem, video podsistem i tvrdi disk sa mogu programirati da se automatski ugase nakon određenog vremena neaktivnosti, bilo zajedno, bilo odvojeno, sve u koracima od po jedan minut. Ako vam se to ne dopada, a znate da su dva najveća potrošača u ovakvim mašinama prikaz i tvrdi disk, možete dodatno invertovati prikaz, pa umesto teget slova na svetloj pozadini imati svetla slova na teget pozadini. Kombinovanjem inverzije prikaza i gašenja prikaza i tvrdog diska nakon 3 minuta neaktivnosti, radni ciklus akumulatora se produžuje i te kako.

Što se punjenja tiče, ono može biti sporo (ako ujedno i radite na mašini) ili brzo (ako je mašina isključena): u slučaju brzog punjenja potreban vremenski period je oko tri časa. Iako ovo nije baš mnogo brzo, bolje je od forsiranog punjenja za manje od sat vremena, jer omogućava duži radni vek akumulatora. Po nama, ovo je dobar kompromis između modernog shvatanja brzine i starog načina rada, kad koga je za punjenje trebalo 9-12 časova.

ŽIVOT SA N300

Ukupna težina celog paketa (računar, ispravljač, kablovi i torba) iznosi oko 3.7 kg (sam računar sa akumulatorom 2.8 kg). To je veoma malo u poređenju sa težinom naše dosadašnje referentne mašine za prenosnike – mašine *Chicony LT3600*, koja ima konst procesor 80286, a je teška nekih 7 kg sama za sebe (odnosno oko 8.5 kg sa kablovima, ispravljačem i torbom).

Iako je u celini odlično konstruisana, ova mašina ima i nekoliko mana. Od relativnih mana, vedma je nezgodno to što je ugradnja numeričkog koprocesora prilično komplikovan posao: jeste da se to ne radi svaki čas, ali videli smo i daleko bolja rešenja. Druga relativna mana je to što se memorija mora širiti samo posebnom memorijskom karticom a ne standardnim delovima: verujemo da je ovo drugo rešenje i odlije mogućo, mada priznajemo da nije baš lako ostvarivo. Treća relativna mana se odnosi na raspored kursorskih tastera: nisu problem strelovi nagore i nadole, ali bi trebalo da tasteri udesno i nadole zamene mesta kako bi raspored odgovarao standardnoj tastaturi. Posle mesec dana rada, još uvek nikako da se naviknemo na raspored ova dva tastera.

Jedina apsolutna mana koju podvlačimo jeste činjenica da poklopci odeljka za akumulator, a ređe i poklopac odeljka za dopunsku memoriju, umeju da se olabave, pa čak i da se potpuno odvoje. Smatramo da se ovo moglo bolje rešiti i čudi nas da je to promaklo Japancima, kao proizvođačima poznatim po preciznosti.

Srećom, vrlina je daleko više. Prvo, korisnik će biti zadovoljan kvalitetom i izvedbom prikaza na računaru: ovo je jedan od najboljih koje smo od sada videli. Osvetljenje je izuzetno ravnomerno, tako da se okom uopšte ne primećuje da je osvetljenje prikaza ivično a ne pozadinsko. Sem toga, efekat "curenja" ("bleeding") je sveden na nivo koji od sada nismo videli – gotovo da ga uopšte i nema, sem u slučaju grafičkih prikaza kada se ipak pomalo primeti. Veliki pomak napred u odnosu na starije generacije.

Tastatura jeste membranska, ali ostavlja utisak velike solidnosti, a osećaj prilikom upotrebe je veoma dobar, pa i odličan (mada je ovo po definiciji veoma subjektivna stvar). Raspored tastera je u celini odli-

Kontakt adresa

TECHNOTRON
YU-35000 Svetozarevo
Tel./Fax. (035) 233-671

KROZ USKO GRLO

Usko grlo u svakom računaru je sva ono čemu procesor mora da pristupi posredstvom sabirnice. Ona je kod ISA standarda spora, a kod mikrokanala ili EISA standarda teorijski brza, ali u praksi nije sve onako kako se priča. To otvara vrata posebnim rešenjima, a mi vam prikazujemo jedno od njih, i to u dve verzije.

Nije tajna da se unutar računara najsporije pristupa onim periferalima koji su vezani preko sabirnica. Stari ISA standard je vezan za 8 MHz sa jednim stanjem čekanja, što u kontekstu procesora 80486DX koji radi na 33 ili 50 MHz zaista predstavlja totalni apsurd.

Dejan V. Veselinović

SAOBRAĆAJNA GUŽVA

Novi standardi, IBM-ov mikrokanal i EISA, obećavaju daleko veće brzine, sa prenosom i do 33 MB/s, odnosno 4-5 puta brže no ISA standard. Međutim, uprkos zalaganju IBM-a i grupe veoma cenjenih proizvođača za ove nove standarde, oni su ipak ostali manje-više puka obećanja. Najbolji dokaz je pojava drugog rešenja, odnosno pristupa celom problemu, u vidu lokalne sabirnice (local bus).

Teorija, a na žalost i praksa, su veoma jednostavni. U kontekstu najbržeg mogućeg pristupa spoljnoj memoriji, današnji procesori su doslovno stotinama i hiljadama puta brži (bar interno), to znači da procesor provodi veoma mnogo vremena radeći uprazno i čekajući da dobije tražene podatke. Pošto za sada još uvek radimo sa rednom (fon Njumanovom) arhitekturom, to znači i da praktično ceo računar stoji u mestu i čeka odziv prozvanog periferala. Svaki od navedenih standarda se obraća ovom problemu na drugačiji način, ali sa istom filozofijom – skratiti vreme čekanja. Bez obzira za koji se opredelite, u osnovi vam se obećava skraćenje vremena čekanja za bar 3:1, a nerelko i mnogo brže.

Sve je to lepo i fino, ali reći ćete vi, šta je sa postojećom bazom opreme? Treba li je sada pobacati kroz prozor i kupiti sve iznova? I najzad, pošto je sva ta lepota od brzine?

Sve to košta jako mnogo, ali ni to nije glavni problem. Osnovni problem se vrti oko raspoloživosti odgovarajućih periferala, drugim rečima, njih i nema tako mnogo, a kada ih ima, obično su veoma skupi, čak i sami po sebi podrazumevaju drugačiju osnovu i najzad, po našem iskustvu, nisu se proslavili efektima.

I eto prostora za jedno lepo tržišno razmišljanje: prvo pitanje je šta prvo i/ili najviše treba ubrzati, drugo, kako to uraditi da bude interesantno što širem krugu potencijalnih korisnika, a treće i poslednje, kako sve to uraditi za što manje para?

Dva aspekta rada sa računarima prosto vape za ubrzavanjem: video i disk podsistemi. Od ta dva podsistema onaj sa diskom ima prioritet jer je posle flopi-disk jedinica ubedljivo najsporiji deo računara. Spori MFM sistem je zamenio brži RLL, a njega još brži IDE standard, sve to nije dovoljno dobro, iako je vreme direktnog čitanja diska sa početnih 100 mS (Seagate ST-225 u originalnoj verziji) palo na ispod 10 mS (Fujitsu M2624T). I tako je rođen keš kontroler.

MALO TEORIJE ...

Keš kontroler diska je zapravo čitav računar smešten na jednu pločicu: starije generacije pločica su imale punu dužinu, a novije su već skraćene na dvotrećinsku dužinu (integracijom, odnosno smeštanjem niza diskretnih integrisanih kola u jedno veliko). Poseduje sopstveni procesor i memoriju, a, ako je dobro napravljen, i veliki izbor naknadnih podešavanja. Naravno, veza sa diskom sa jedne i matičnom pločom sa druge strane ne sme biti kompromitovana na bilo koji način.

Od procesora, najčešće se koriste modeli firme Intel, mada ima ovakvih kontrolera i sa procesorom Motorola 68000. Intel ima prednost iz dva razloga. Prvo, Intel je oduvek propagirao logiku maksimalne hardverske integracije, što je za ovakve potrebe mnogo bolje nego upotreba procesora sa labavijom hardverskom integracijom i boljom otvorenošću ka programeru (suprotno od logike pravljenja računara), i drugo, Intel poseduje neprevaziđen niz opštih i posebnih procesora, opet na nizu radnih taktova, a sve to po izuzetno povoljnim cenama. Dakle, Intel bez konkurencije.

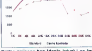
U okviru Intel-ove porodice, najčešći izbor čine procesori 80188 za 16-bitnu sabirnicu, 80186 za 32-bitnu sabirnicu, a u novije vreme i procesor 80386SX za 32-bitnu EISA sabirnicu (Mylex). Usled zaista smešne cene procesora (oko USD 5 za 1 000 komada), niko ni ne ide ispod radnog takta od 12 MHz, 16 nije retkost, a već se javljaju i taktovi od 20 MHz u slučaju kontrolera sa procesorom 80386SX.

Zašto baš hibridni (istina, visoko integrisana verzija, ali ipak hibridni) 16/8-bitni procesor? Zato što je jeftiniji za implementaciju, a manje je zahtevan po pitanju memorije, pošto se zadovoljava i sa samo jednim SIM modulom. Naravno, i najjeftiniji je.

Fleksibilnost obuhvata dva aspekta. Prvi se odnosi na vezu sa procesorom i upravljačkim sistemom, u idealnom slučaju, kontroler bi trebalo da se prilagođava potrebama svih matičnih ploča (totalna transparencija) i svim iole važnijim upravljačkim sistemima, tako i samo tako postaje moguće iskoristiti sve potencijalne prednosti ove tehnologije. Drugi aspekt se odnosi na upotrebljeni algoritam u smislu vrste i podešljivosti prema raznim specifičnim situacijama, opet, samo tako je moguće smatrati da se radi o zaista univerzalnom uređaju.

Od raspoloživih algoritama u praksi se najčešće koriste dva: prvi se zove *write through*, a drugi *write back*. U prvom slučaju, memorija na kontroleru se koristi kao neka vrsta amortizera, sve što na disk treba zapisati se odmah primi u memoriju, ali proces pisanja otpočinje odmah i potpuno je neselektivan, što znači da se na disk zapisuje celokupna datoteka. U drugom slučaju (*write back*), postojaće vremenska zadrška fizičkog pisanja na disk, a pisaće se samo izmenjeni delovi datoteka, a ne cele datoteke.

Slika 1



Brzine prenosa bez (donja kriva) i sa (gornja kriva) keš kontrolerom

Ta vremenska zadrška može biti ili jasno definisana po trajanju (a nakon isteka zadatog perioda doći će do pisanja ma šta računar tog trenutka radio, što znači i mogući prekidi trenutnog posla), ili pak može biti klizna. Kada je klizna, to znači da će se fizički zapis odložiti do nekog momenta kada je procesor slobodan, pa će se zapis obaviti tek tada. Drugim rečima, do maksimuma se koristi cela koncepcija, jer je centralni procesor rasterećen po dva osnova: samo pisanje se za njega svodi na prosto izdavanje komandi, nakon čega kontroler preuzima dalji posao, dok se fizičko pisanje se obavlja iz memorije na kontroleru, pa je i sam proces davanja povratnog signala centralnom procesoru drastično smanjen.

Jedna dopunska prednost dobro napravljenog keš kontrolera je rasterećenje sistemske memorije od zahteva za keš memorijom, ali i od potrebe da se koristi neki surogat za ubrzanje rada sa diskom u vidu bilo kakvog programa za keširanje. Druga dopunska prednost je eventualan niz mogućnosti, kao što su preslikavanje diskova (*mirroring*) i/ili sabiranje kapaciteta, koje bez ovakve tehnologije jednostavno nisu moguće.

Najzad, celom problemu keširanja tvrdog diska treba dodati i problem usaglašavanja sa maltene bezbrojem naizgled sličnih ali realno veoma različitih platformi i priličnim razlikama u pristupu razaznavanja samog diska i vrste veze od strane različitih upravljačkih sistema. Nije to baš jednostavno, ali može se.

... I MALO PRAKSE

Ispred nas su se našla dva keš kontrolera, za koje smo ubrzo shvatili da potiču iz istog izvora, ali da pripadaju različitim generacijama, i to ne samo u vizuelnom smislu.

Oba modela koriste procesor Intel 80188 na 12 MHz, a najveća brzina prenosa prema deklaraciji iznosi 5 MB/s. Oba mogu da prime do 16 MB keš memorije, ali ujedno mogu da rade i sa svega 256 KB memorije, odnosno jednim SIM modulom. Moduli se mogu mešati po želji, koristeći module sa 256 KB, 1 MB i 4 MB, računajući i mešanje na istoj kartici.

Oba modela imaju isti sistem za podešavanje koje se obavlja na potpuno isti način, a to je izmenom parametara u sopstvenom SETUP programu, koji se potom zapisuju u EPROM na pločici. Gotovo sve ponuđene mogućnosti su istovetne, sem jedne: novija generacija kontrolera nudi izbor između sabiranja kapaciteta dva diska povezana na red i preslikavanje sadržaja jednog na drugi, dok starija generacija uopšte ni ne nudi mogućnost preslikavanja.

Kontroleri primaju ukupno po četiri diska, dva i dva u granama za "C" i "D" disk. To mogu biti bilo kakvi IDE diskovi, od bilo kojeg proizvođača, bilo kojih osobina i bilo kojih kapaciteta: ovu smelu tvrdnju smo prvu provenili i zaista, po prvi put u našoj praksi smo bez ikakvih problema povezali Quantum sa Fujitsu, Conner, Seagate i Western Digital diskovima,

kao i bilo koji do njih se bilo kojim drugim. Time je dokazano da je rak rana IDE "standarda" međusobno povezivanje diskova iz različitih izvora, konačno rešena hvalim te Bože!

C1 i C2 diskovi se po želji mogu sabrati u jedan pri čemu je zbir jednak pojedinačnim kapacitetima. Dalje, kontroleri omogućavaju premošćivanje maksimuma do 556 MB koje priznaje DOS 5.0, tako da možete povezati bilo kakve diskova do kapaciteta do 4 GB po disku. No, čak i bez toga, povežite samo četiri Fujitsu diska od po 512 MB i već imate ukupni kapacitet od preko 2 GB – a to je već strogo teritorija mini računara. Ako vam ni to nije dosta, znajte da u svakoj mašini možete imati po dva ovakva kontrolera, što vam opet duplira maksimum kapaciteta; pošto se već najavljuju IDE diskovi do po 650 MB, ukupno možete raspolagati sa 5.2 GB, što je već klasa midi računara. Čak i ako se opredelite za verziju sa maksimalnom bezbednošću, što znači sa preslikavanjem, opet imate na raspolaganju 2,6 GB preslikanih podataka.

Podešavanje nudi čitav niz opcija. Možete podesiti kontroler da radi za *DOS/Windows OS/2 UNIX XENIX* i *Novell*. Možete koristiti oba opisana metoda pisanja na disk, a ako želite, možete potpuno isključiti keš memoriju. Svaki disk u sistemskom BIOS-u prijavljujete kao tip 1, čime se obezbeđuje transparentnost za sva upravljačke sisteme (taj tip priznaju i prepoznaju svi). Sve ostale podatke kontroler uzima sa samog diska. Možete podesiti veličinu niza (4, 8 i 16 KB), broj blokova koji će se unapred učitati (0-3), gde će se nalaziti BIOS kontrolera (ROM ili RAM), vrstu algoritma (*write back* ili *write through*), vreme do zapisa (0-15 s i beskonačno), itd. Naravno, možete podesiti i razne stvari oko diska (sabiranje, preslikavanje, veza za diskove veće do 556 MB itd).

Jedini stvarni posao koji imate jeste da izvadite svoj dosadašnji IDE kontroler i na njegovo mesto stavite novi. Zatim još morate da kablove sa diska i flopi disketa prebacite na novi kontroler. I to je sve.

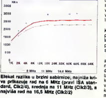
Ako se, kojim slučajem, zbog neke egzotične kartice, desi da je adresni prostor koji vam treba već zauzet, ni to nije problem. Lako ćete kratkospojnicima prebaciti kontroler na neku drugu adresu.

PERFORMANSE

Keš kontroleri se mogu odnositi prema performansama na jedan od tri suštinska načina: mogu ubrzavati fizički pristup disku, mogu poboljšavati brzinu prenosa ili koristiti neku kombinaciju prve dve mogućnosti. Oba kontrolera sa kojima smo mi imali posla koriste treću mogućnost.

Kao test poligon upotrebili smo jednu platformu sa procesorom 80386DX na 33 MHz, sa 64 KB keš memorije i diskom Seagate ST-3144A; ovo je sasvim popularan disk, dobro poznat na našem tržištu, pa niko ne može reći da se radilo o nekoj teškoj egzotici (naprotiv!). U izvornoj verziji, mašina koristi

Slika 2



Efekat razlike u brzini sabirnice, najniža kriva prikazuje rad na 6 MHz (pravi ISA standard, Clk2/4), srednja na 11 MHz (Clk2/3), a najviša rad na 16,5 MHz (Clk2/2)

jedan Longshina IDE kontroler, čija je jedina osobenost izuzetna otpornost na brzinu sabirnice, što nam je omogućilo da sabirnicu umesto na 8 MHz držimo na 11 MHz. Isti kontroler nije mogao da radi na većoj brzini (16.5 MHz), dok su oba keš kontrolera mirne duše radila i na većoj brzini. Ipak, za potrebe testa, ograničili smo se na nestandardnih 11 MHz (Clk2/3) jer je tako sve radilo i bez izmena, a svi delovi su dobro provereni na tim brzinama.

Prvi test se odnosio na proveru brzine prenosa bez i sa keš kontrolerima. Pokazalo se da se dva kontrolera ponašaju maltene identično, a razlike se mogu otpisati na varijacije u rezultatima testova, pa zato govorimo o "kontrolerima" a ne o ovom ili onom kontroleru. Elem, slika 1 prikazuje brzine prenosa bez (donja kriva) i sa (gornja kriva) keš kontrolerom. Nije došlo do nekog značajnijeg povećanja brzina prenosa kao takve, ali jeste došlo do veoma osetnog izravnanja "brda i dolina" koje nastaju kada traženi blokovi prerastu segmentiranu keš memoriju na samom disku.

Slika 2 prikazuje efekat razlike u brzini sabirnice, najniža kriva prikazuje rad na 8 MHz (pravi ISA standard Clk2/4), srednja na 11 MHz (Clk2/3), a najviša rad na 16.5 MHz (Clk2/2). Iako nekih razlika ima, posebno između 6 i 16.5 MHz, one nisu ni blizu razlici u apsolutnim brojkama i iznose 31% u najboljem slučaju. U praksi stvarne razlike u merenjima na testu pretraživanja baze podataka su potpuno zanemarive jer iznose jedva 2-3%.

Slika 3 i 4 prikazuju ponašanje računara na testovima sa malim (200 B) i velikim (2 KB) datotekama, složenim u nizove od 64KB do 3.8MB. Interesantno je primetiti ujednačenost brzina rada diska samog za sebe, dok je rad sa keš kontrolerom daleko manje ujednačen. U stvari, radi se o logičnom i očekivanom efektu: razlika u brzini rada je velika sve dotle dok je količina memorije na kontroleru veća ili jednaka veličini zahteva. Čim zahtev postane veći od raspoložive memorije, efekti brzine se osetno smanjuju, to se jasno uočava posebno na testu sa velikim datotekama. Razlika u brzini je velika sve do veličine datoteka do 1.97 MB, što je veoma blizu instalirane memorije do 2 MB.

Najzad, slika 5 prikazuje rezultate testova iz paketa *Unitest* a radi se o testovima rada sa bazom podataka. Kao što se vidi, razlike su najveće u testu indeksiranja (gotovo 11 puta brže) i testu indeksnog pretraživanja (5.1 puta brže). Uzgred, ovo je jedina slika sa logaritamskom skalom.

Toliko o testovima. Međutim, subjektivni stav srećnika sa ovakvom karticom će biti daleko povoljniji od prikazanih brojki, jednostavno, efekat je takav da disk postaje munjevito brz, posebno prilikom učitavanja i pisanja na disk, naravno, sve dotle dok na kartici ima dovoljno memorije. 2 MB je sasvim dovoljno za veliku većinu aplikacija i potreba u slučaju korisnika-pojedinca. Za mreže, odnosno u serverima, treba obezbediti 512 KB po korisniku, a ako je baš tesno, moći će i 256 KB po korisniku; oba brojka su dobijena čisto empirijski. U prvom slučaju, 16 MB je dovoljno za 32 korisnika, a u drugom i do 64 korisnika, naravno da može i više, ali to će ipak postati malo tesno u situacijama kada veći broj korisnika postavi niz zahteva za kapacitetima servera.

U slučaju korisnika-pojedinaca, ovakva kartica sa 2 MB će biti sasvim dovoljna u najvećem broju slučajeva. Potencijalno zainteresovani korisnici bi mogli biti svi oni čiji posao traži dosta obraćanja disku, dobar primer bi bili programeri, profesionalci u oblasti DTP-a, ljudi koji razvijaju baze podataka i drugi.

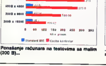
SKUPO... ALI VREDI

Model Galaxy DC-600 i Longshine LCS-6641 su u osnovi isti proizvod, uz samo dve razlike: prvi pripada tehnološki starijoj generaciji a drugi novijoj, integrisanijoj. Druga razlika, koja nije površna, jeste činjenica da noviji model pruža mogućnost preslikavanja sadržaja jednog na drugi disk, mada uz gubitak mogućnosti sabiranja kapaciteta dva diska; u nekim slučajevima, to može biti značajna prednost. Međutim, u praktičnom smislu, nema razlike između ova dva modela.

Praktične dobiti do ovih uređaja, koje su nesumnjive i nisu male, moraju se staviti u kontekst potreba se jedne i cene sa druge strane. Pretpostavljajući da sami svoje potrebe najbolje poznajete, ostaje nam da razmotrimo cenu. Trenutno, ova dva uređaja se prodaju kod nas za oko DEM 500 (ili ekvivalent u dinarima, ali prvo pitajte pošto je čija marka jer su razlike velike), čemu treba dodati cenu 2 MB memorije, odnosno daljih DEM 150.

Za sada, naš savet glasi: ovo su dobri uređaji, tamo gde je potrebno mogu odlično obaviti posao, ali imajući u vidu cenu, ipak dobro razmislite. Ali ako imate te pare, ne razmišljajte – kupite.

Slika 3



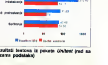
Ponašanje računara na testovima sa malim (200 B)...

Slika 4



...i velikim (2 KB) datotekama, složenim u nizova od 64KB do 3,8MB

Slika 5



Rezultati testova iz paketa *Unitest* (rad sa bazama podataka)


ComTrad
the art of computer making
Procesor 386/40MHz
Cache 64kB, 4MB RAM
Floppy 1.2MB i 1.44MB
Tvrdi disk 107MB
SVGA grafička karta 1024x768/1MB
SVGA kolor monitor
2 serijska i 1 paralelni port
Mini tower kućište
ASCII tastatura
Miš
2,990.-
ComTrad YU, Beograd
Genex apartmani, Apt. 20 - V. Popovića 6 - Tel. (011) 222-26-52, 222-41-51
Fax (011) 222-41-39 - Radno vreme: 9-17, ponedeljak-petak
MANIR
Ruma
Grobljanska 2
Infotrade
Priština
Hekurudha 7
Tel (038) 25-830
Fax (038) 25-822
DUGA COMPUTER
Beograd
Roze Luksemburg 7
Tel (011) 587-468
Fax (011) 587-468
COMTRAD
KRAGUJEVAC
Kragujevac
Kragujevačkog oktobra 102
Tel (034) 60-336
Fax (034) 57-117
ComTrad
COMPUTER



IMTEL 486-33MHz
IMTEL 486-33MHz
za samo
1999 dm
486SX-33MHz
1Mb RAM
40Mb HDD
1.2Mb FDD
SVGA mono
256kB
VGA mono
monitor
486 procesor za nove operativne sisteme
WINDOWS na 486
IMTEL
Vam obezbeđuje
WINDOWS 3.1
486 + S3
superbrzu
grafiku
32000 boja
135-420
486 procesor
- minimalno početno ulaganje manje nego za 386DX
- za samo 5 minuta možete nadgraditi svoj računar u 486DX-33 486DX-50 ili 486DX2-66
- doplata samo za razliku u ceni
IMTEL 486 je računar koji raste zajedno sa Vama
BENCHMARK 486SX-33MHz
LANDMARK SPEED
brzina
MHz
300
200
100
486DX2-66
486DX-50
486DX-33
486SX-33
cena
DM
1000
2000
3000

INTELOV KVANTNI SKOK

Samo godinu dana pošto je Intel (aprila 1989.) predstavio svetu svoj 80486 procesor, počela su prva nagađanja o njegovom nasledniku pod tajanstvenim imenom P5. Danas se o novom procesoru zna već gotovo sve.

Nebojša Novaković



Intel 486 je bio zamišljan kao procesor koji će se po brzini nalaziti u samom vrhu 32-bitne elite, čuvajući pri tom potpunu kompatibilnost sa zastarelim 16-bitnim DOS svetom, sa hiljadama njemu dostupnih softverskih paketa kao i, da ne zaboravimo, milionima sati učenja uloženih u taj sistem. 486 je po Intelovoj propagandi krajem 1989. trebalo da bude brži čak i od Motorola 88100, tadašnjeg RISC brzinskog šampiona. Intel je i napravio procesor brži od MC 88100 – međutim, to je bio svima znani 860 RISC. 486 sa svoje strane ne samo da je zaostajao za 88100 nego i za naposlednim konkurentom MC 68040. Tadašnji vrh 68000 porodice, koji je na manje megaherca bevao bolje performanse zahvaljujući zdravijoj arhitekturi bez tereta kompatibilnosti sa prestorijskim 8-bitnim akumulatorcima iz 1972, imao je još jedan adut koji Intel tek odnedavno primenjuje u novom 486DX2 – unutrašnji takt dvostruko viši od spoljnog. Tako 33 MHz 68040 svoje unutrašnje operacije zapravo obavlja na 66 MHz!

486 nije ispunio ni postavljeni cilj da na istom taktu u proseku bude tri puta brži od 386, uzimajući u obzir iste veličine spoljnog keša i ostale stvari koje su u sistemu identične. Ubrzanje se kreće u proseku oko 70% za celobrojne i 2.2 puta za FP aplikacije u odnosu na 386 - 387 par. Stvarne performanse procesora 486 su iznevarile Intelova obećanja.

Već tada je Intel bio u toku razvoja 586, planirajući njegovo lansiranje za kraj 1991. a prodaju najkasnije na proleće 1992. Sve karakteristike koje ćemo ovde navesti i željeni nivo performansi bili su postavljeni već tada, tako da tehnologije nije predstavljala teškoću. Međutim, postavio se drugi ozbiljan problem – kako spojiti zastarelu 8086 i DOS kompatibilnost i nove RISC tekovine, kao što su superskalarnost (obavljanja više naredbi po taktu), Harvard arhitektura (odvojen pristup, negde i adresni prostor, naredbama i podacima), multiprocesorski rad itd. Jednostavno, stari DOS softver teško se mogao uklopiti u takvo okruženje. Rešavanje ovog problema, čija je srž ujedno i najveća slabost 80x86 porodice – teret kompatibilnosti sa zastarelom prošlošću – odnelo je još godinu dana. Šta nam onda 586 donosi?

Prva i najprimetnija novina je ime. Naime, P5 se po svoj prilici neće zvati 586, zbog nemogućnosti da se broj kao ime zaštiti od kopiranja sve brojnijih klonera 32-bitnih 80x86 procesora. U Intelu je završen interni konkurs za izbor najpogodnijeg imena (tako je odabran i naziv za *OverDrive*). Moglo bi to da bude, na primer, *TurboInjection* ili nešto slično.

Druga očigledna novost je veće kućište – 19x18 PGA kao kod 860XP. To nije jedina sličnost sa 860XP. Spoljne magistrale 586 i 860XP su identične i po strukturi i po tajm-rezima, a takođe i upravljanje memorijom i mehanizmi očuvanja ispravnosti sadržaja keša. Tako 64-bitna 66 MHz spoljna magistrala kod 586-66 ili 860XP-66 dostiže 426 Mbajta/s sa najbržim 2-1-1-1 *burst* ciklusima iz keša drugog nivoa. To je četiri puta brže od 486DX2-66 sa njegovom 33 MHz 32-bitnom spoljnom magistralom. Ista magistrala kao kod novih 860 procesora znači i veoma lako izvodljiv paralelni rad više 586 i 860 procesora na zajedničkoj memorijskoj magistrali.

Treća novost je gotovo (ali nikako sasvim) potpuna RISC arhitektura, sa malim mikrokod CISC delom za one složenije 486 naredbe koje je teško implementirati u hardveru. Usko vezana za to je superskalarnost u 586. Kao što znate, superskalarnost predstavlja mogućnost lansiranja više mašinskih naredbi u svakom takt-ciklusu. To je postignuto tako što keš za naredbe kod 586 ima širinu od 64 bita, dovoljnu da se učitaju dve proste 32-bitne naredbe u svakom ciklusu, a instrukciona jedinica odlučuje da li se oba učitane naredbe mogu izvršiti paralelno ili ne. Obično se proste instrukcije lako paralelizuju. Dok 486 ima jedan pajplajn u pet nivoa, 586 ima dva paralelna pajplajna – dve celobrojne jedinice sa svojim ALU koje rade u paraleli. Naravno, prva naredba u paru ne sme biti skok. Kako grananja znaju da poništi ceo sadržaj pajplajna, 586 ima poseban bafer za grananja koji je u biti neka vrsta prognozera za najverovatnija grananje.

586 i dalje poštuje matori 80x86 registarski model sa 8 registara, od kojih se prvi u ranijim programima koristi kao akumulator. Kad pogledamo današnje prave RISC procesore kao MIPS R4000, MC 88110 i DEC Alpha 21064 sa po 32 registra, ili SuperSparc sa 136, ovo zaista izgleda bedno. Bez obzira na mehanizme koje Intel koristi da poveća efikasnost ovih registara, rezultati ne mogu da se porede sa RISC mašinama koje to imaju od samog početka u svojoj arhitekturi. Ista stvar pogotovo za puni 64-bitni Alpha procesor. Važi i za superskalarnost, multiprocesorski rad, *write back* keš itd. Sva savremena RISC arhitekture već odavno su ušle u te štosova.

Stari 8-registarski model (pazite, još uvek nisu u pitanju konačne informacije – Intel je verovatno dodao neka registra specifične za 586 arhitekturu!) važi i za ugrađeni FP procesor, koji je dramatično unapređen u odnosu na onaj stari u 486-ici. Suština je u tome da je prenet dobar deo brze arhitekture 860 FP jedinice sa mogućnošću lansiranja (pazite, ne baš i izvršavanja!) po jedne FP naredbe u svakom taktu. Taj potpuno novi FP pajplajn u 8 nivoa sam donosi i više nego dvostruko ubrzanje. Druga komponenta je znatno ubrzano vreme izvršavanja svih naredbi, posebno transcendentalnih, koja je, zahvaljujući novim algoritmima i hardverskoj bazi, ubrzenja i više od 20 puta kraća u odnosu na 486DX. To znači da će sve matematičke aplikacije sa 586 i njegovim FP sabiračem, množačem i deliteljem doživeti strahovito ubrzanja.

Jedan od elemenata anuliranja malog registarskog seta je i velika keš memorija. 586 ima dva odvojena keša: 16 KB keša za naredbe i 16-KB *dual-port* data keša (paralelno mogu da pristupe dve jedinice) za podatke. Oba keša imaju efikasnu četvorostruko set asocijativnu arhitekturu, a koriste i "write-back" način, koji se delimično isključuje kada imate spoljni keš drugog nivoa.

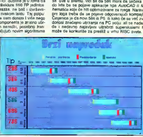
586 inače ima preko 3 miliona tranzistora – 2.5 puta više od 486. Najveći deo, oko 2 miliona, zauzima ugrađeni keš. Uzput, i potrošnja je veća – 10 W na 66 MHz – pa je hladnjak zato obavezan.

586 sa svojim poboljšanjima prodire u sferu snažnih RISC procesora. Ali, zbog ne baš sjajne arhitekture ranijih procesora s kojima kompatibilnost mora biti sačuvana, svi najnoviji RISC procesori su jači: i SuperSparc i MIPS R4000 i MC 88110 i IBM POWER i, posebno, DEC Alpha.

Intel tvrdi da se 586 bez problema nosi sa MIPS R4000 i Texas/Sun SuperSparc čipovima. Možda će tako i biti kada operativni sistemi kao *Windows NT* i aplikativni softver budu optimizovani za njega. Međutim, ako uzmemo u obzir trenutni 32-bitni DOS, OS/2 i UNIX softver, kao i činjenicu da je na superskalarnim RISC procesorima dosta softvera već optimizovano za njih, dobijamo sledeću tabelu gde su vrednosti za 586 očekivane, ne potvrđene. Svi procesori podrazumevaju 1 MB najbržeg keša drugog nivoa, tako da je memorijska magistrala manje bitna. Test SPECmark 89, čije rezultate navodimo, je relativno objektivan agregatni aplikativni test, koji uzima u obzir i celobrojne i FP performanse. Njegovi rezultati se s vremenom menjaju, ponajpre zbog optimizacija kompajlera. Performanse su date u odnosu na VAX 11/780 i obično su 10 – 20% niže od VAX MIPS vrednosti. Naravno, svi ovi procesori, izuzev DEC 21064 Alpha, sigurnim koracima idu ka frekvencijama reda 100 MHz i dvostrukom internom taktu, i to važi i za 586. Kod 64-bitne Alphe, sa već ultravisokim taktom, naglasak će pre biti stavljen na dalju internu paralelizaciju, za koju je veoma pogodna.

procesor	takt	int keš	SPECmark
MIPS R4000MC	66	8i + 8d	90
TI SuperSPARC	50	20i + 16d	80
DEC 21064-AA	100/200	8i + 8d	175
Intel 486DX2	33/66	8id	20
Intel P5	66	16i + 16d	45

Za 486DX2 se podrazumeva 256 KB spoljnog keša. Sve u svemu, vidi se da 586 mora da sačeka bar do leta da se pojave aplikacije tipa *AutoCAD* ili *Mathematica* koje će biti optimizovane za njega. Naravno, pre toga treba da se pojave odgovarajući kompajleri. Činjenica je da novi 586 ili P5, ili kako će se već zvati, donosi značajno ubrzanja na PC polju, ali se nadamo da i nedavno najavljeni ultrabrzi superskalarni 860 može da konkuriše za prestiž u vrhu RISC sveta.



Loša vest za klonera. Pošto više timova radi paralelno na razvoju novih procesora, vreme između pojava novih generacija je sve kraće

KROZ PROZOR PO SVAKU CENU

Eh, kako je nakad bilo lako programirati: imali smo jednog proizvođača kompajlera i jedan paskal – sva razlika u izgledu i kvalitetu programa su postojala samo zahvaljujući (na)sposobnosti programera. Sada imamo mnogo više raspoloživih jezika, na tona dokumentacija „koja se mora pročitati da bi se počelo sa radom" i na hiljada stvari na koja programer mora da misli – ukratko, dobrodošli u svet *Windows* programiranja!

Dušan Popović

Uz ozbiljan operativni sistem kakav je *Windows*, neophodan je i ozbiljan razvojni sistem za programe što je bilo jasno i proizvođačima programskih jezika. Razlika (to je blaga reč) je u pristupu raznih softverskih firmi rešavanju tog problema. Prvi je pokušao Microsoft, što ne znači i da je uspeo u tome. Paket *System Development kit* je samo dobar vic na temu programiranja za *Windows*. Tako glomazan i gotovo neupotrebljiv paket nije skoro viđen u grupi programerskih alata.

Od tada je prošlo oko godinu dana i velike softverske firme su imale vremena da se malo bolje organizuju, da pročačkaju šta treba programerima i kako jedan *Windows* programerski paket treba da izgleda. U kratkom roku smo doživeli lansiranje više jezika za razvoj programa pod *Windows*-om: *Vizual Basic, Borland C++ 3.0, Quick C for Windows, Object Vision*, i, najzad, *Microsoft C++ 7.0*.

Kada kupujete jedan programski jezik morate biti svesni svojih potreba. Naravno, od viška mogućnosti glava ne boli, ali ako ste na prosečnom budžetu dobro ćete razmisliti da li su vam baš sve mogućnosti nekog profesionalnog paketa potrebne. Toga su svesni i proizvođači softvera, pa u ponudi imaju profesionalne verzije paketa i one druge, okrnjene za po neku pogodnost.

Ako pogledamo trenutnu ponudu C i C++ programskih jezika za *Windows* okruženje, vidimo da tu dominiraju dva proizvođača – *Borland* i *Microsoft*. Čak i podela paketa na „profesionalna" i „ona druge" izgleda namerna: oba proizvođača nude po jednu profi i jednu amatersku verziju. Koliko su „profi" verzije stvarno profi, a koliko su „manje profi" verzije stvarno različite od onih drugih, videćemo i na primeru paketa koga prikazujemo. Pošto smo u jednom od prošlih brojeva prikazali *Quick C for Windows* proizvod *Microsoft*-a, red je da vidimo i odgovor od strane *Borland*-a – *TURBO C++ 3.0 FOR WINDOWS*, kompajler koji liči na starijeg brata (*Borland C++3.0*), ali ima malo manje mogućnosti, pa samim tim manje i košta. Već se zna da je Borland uveo novo pravilo u nazive svojih proizvoda: sada se verzije C-a i paskala sa nazivom *Borland* vode kao profesionalne, a sa nazivom *Turbo* kao obične. Pa da vidimo šta Borland smatra dostojnim jednog programera-amatera.

STROGO ZA WINDOWS

"For Windows" nam kazuje da se radi o verziji koja radi samo pod *Windows*-om i samo za *Windows*. Čak se ni bilo kakav DOS program ne može napraviti – ovaj C pravi samo *Windows* EXE i DLL programe. To rezultuje manjim brojem biblioteka i manjim zauzećem mesta na disku (koliko manjim, videćemo). Idealno za programere koje razvoj programa za DOS ne zanima.

Kutija u kojoj dobijate *Turbo C++3.0 for Windows* nije ni malo portabl – ne izgleda velika, ali je podosta teška. *Borland look* (a zar to treba reći) dozvoljava čoveku da pari oči i pre otvaranja, jer su reklame na poleđini više nego atraktne. Kada se otvori, vidi se zašto je teška – gotovo sav prostor je popunjen knjigama, toliko popunjen da sadržaj morate istresti (!) napolje. A kada se konačno istrese, vidi se da su to jako pristojne i ozbiljne knjige, u što ćete se uveriti iz onog što sledi.

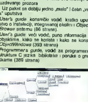
Ono što čoveka najviše zanima kada dobija novi paket je instalacija, tako da verujem da ćete i vi požuriti sa njom. Pošto paket dolazi na 5 AT disketa, može se pretpostaviti koliko će zauzeti na disku. Naša procena je bila oko 12 megabajta (pod pretpostavkom da su diskete kompresovane), tako da smo samouvereno ubacili prvu disketu i iz *Windows*-a startovali *install*. Na to je program nonšalantno izjavio da zahteva 22 megabajta na disku!!! Što je mnogo, mnogo je. Ako uzmemo u obzir da na našim prostorima još uvek postoji gomila diskova od 40 do 50 megabajta, ova cifra zvuči strašno. Odvojiti pola hard diska samo za jedan program!?

Borland očigledno hoće da bude zapamćen i po svojim instalacionim programima – ako se sećate kako se instalira *Turbo Paskal* za *Windows*, onda vam je jasno kada kažem da sada gledate brojčanik na kolima.

Posle pet izmenjanih disketa, kroz oko 45 minuta i sa 22 MB prostora manje na disku, možemo da isprobamo *Borland*-ovo rešenje za zauzdavanje prozora.

Uz paket se dobiju jedno „malo" i četiri „velika" uputstva:

- *User's guide*, korisnički vodič, kratko uputstvo o instalaciji, integrisanoj okolini i *ObjectBrowse* sistemu (86 strana)
- *User's guide*, veći vodič, puno informacija o objektima, kako se korista i kako se koristi *ObjectWindows* (393 strana)
- *Programmer's guide*, vodič za programera, struktura C jezika, biblioteke i poruke o greškama (389 strana)



Povezanost objekata koje koristi program na listingu 1

- *Turbo Debugger user's guide*, uputstvo za korišćenje debagera (287 strana)
- *Resource Workshop*, uputstvo o korišćenju resurs editora, jako korisne alatke (287 strana)

Sva literatura je izuzetno čitljiva i pregledna. Pasusi su označeni ikonama tako da se na prvi pogled vidi o kojoj temi govori pasus: o C++ programiranju, radu sa tastaturom ili o nečem trećem. Sve što je moglo da se organizuje u spisak, organizovano je. Ukratko, naše mišljenje o tome kako Borland pravi dokumentaciju za svoje programe ostaje – jasno i pregledno.

(NIJE) LAKO S WINDOWSIMA

Ukoliko poznajete neki od ranijih Borland-ovih kompajlera (BC++ 2.0, BC++3.0) koji rade pod DOS-om, onda ćete bez problema da počnete rad sa ovim kompajlerom. Svi meniji su isti, čak je i većina *hot key* kombinacija ista. Ipak, primećuje se nedostatak **Debug** menija. Sitan nedostatak, dok vam ne zatreba istervanje bubica iz programa. Naravno, ovaj kompajler ima odvojen debager, ali, kao što ćemo videti, to nije najsrećnije rešenje. Ukoliko imate projekte koje ste radili pod nekom starijom verzijom Borland-ovih kompajlera, možete odmah početi sa radom, jer će *Turbo C for Windows* sam izvršiti sve konverzije. Ako ste mali programe koji su radili pod DOS-om, kako oni uopšte mogu da rade pod *Windows*-om? Lako, uz pomoć biblioteke EASYWIN.

Prelazak sa tekstualnog okruženja kakvo je DOS na grafičko okruženje kakvo je *Windows* zahteva dosta truda – čemu sjajne mogućnosti ako se nijedan već napisan program ne može iskoristiti? Tu opet C razmišlja umesto vas: ukoliko primeti da je program pisan za DOS (tačnije, ako nema *WinMain* funkcije kojom počinje svaki C program pisan za *Windows*) kompajler uključi i biblioteku koja, doduše, sadrži gotovo sve funkcije koje se koriste u DOS-u, ali napisane tako da ceo tekstualni ulaz/izlaz jednog programa ide kroz *Windows* prozor. Rezultat je bio to da je nekoliko obimnih programa napisanih za DOS (pri čemu o *Windows*-u nije bilo ni reči) posle nekoliko pokušaja proradilo u prozoru. Iako se mora voditi računa o par nekompatibilnih ili opasnih poziva (čak su dozvoljeni i **peek/poke**!), uz malo prilagođavanja svaki program će moći da se iskompajlira. Ovo nam je sjajno poslužilo da pod *Windows*-om pokrenemo nekoliko alata koje smo ranije napisali u C-u, koji olakšavaju programiranje, i na koja smo, uostalom, navikli.

OBJEKTNI PROZORI

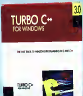
Najvažnije je bilo pitanje da li je ovaj C kompatibilan sa ranijim C verzijama istog proizvođača. Preveli smo nekoliko programa (uz pomoć easywin biblioteke, naravno), od kojih se najveći sastojao od više modula i preko

8000 linija programa. Sve je proteklo u najboljem redu. Onda su zahtevi porasli, pa smo isprobavali primere koji idu uz ovaj C. Samo jedna reč – efektno.

Pristup različitih firmi rešavanju problema programiranja pod prozorima je različit. U *Quick C*-u smo videli kako je Microsoft priložio alat kojim se generiše kompletno stablo programa – **WinMain** funkcija i **case** struktura u koju programer naknadno dodaje delove programa. Da može i kraće od toga, dokazao nam je Borland – ako se malo potrudimo oko objektnog programiranja, možemo znatno smanjiti posao, bar onaj oko dosadnog pisanja uvek istog početnog koda i već pomenute **case** strukture.

Ako neko pretenduje da bude lider u objektnom programiranju kao Borland, onda nije dovoljno što je prvi uveo OOP u svoje jezike već mora uvek biti korak ispred konkurencije i izmišljati nešto novo i neviđeno. Ako već imamo objekte, zašto ne ponuditi i biblioteku objekata, takvu da se odmah može koristiti? Kada pogledate listing, da li biste rekli da je to kompletan editor, sa mogućnostima snimanja i učitavaja teksta, **cut/copy/paste** opcijom i traženjem teksta? Gde je stalo sve to u samo pedesetak linija koda? Pa naravno, u biblioteku objekata, koju Borland naziva *Object Windows*.

Object Windows je biblioteka koja omogućava potpuni red neke *Windows* aplikacije samo uz pomoć objekata. Pri tome su iskorišćene objektne osobine i jednostavnost pri korišćenju objekata, tako da o svim dosadnim detaljima oko grafičkog interfejsa (otvaranje dijaloga i maski za unos, komunikacija i upravljanje prozorima) brine sam C. Time se programiranje za *Windows* približava programiranju za DOS po jednostavnosti, ali ne i po grafičkom izgledu – aplikacije rađene uz pomoć *Object Windows*-a sadrže sve one lepe stvari zbog kojih cenimo *Windows* programe. Naše je mišljenje da (uz malo truda i vremena potrebnog za učenje *Object Windows* poziva) programiranje pod *Windows*-ima postaje lako, što do sada nije bio slučaj ni sa jednim C paketom (ne računajući starijeg brata BC++ 3.0).



BORLAND LOOK

Ako ste se navikli na *Windows* izgled dijaloga i maski za unos, kada startujete ovaj C naći ćete se u čudu – gotovo sve je redizajnirano. Dijalog više ne izgleda kao što je izgledao nego je popunjen rasterom, tasteri (kao OK i CANCEL, na primer) su promenjeni, ukratko, utisak više nije isti. Da li će vam se to svideti ili ne, zavisi od ukusa, ali smo mi lično više naklonjeni starom izgledu. Uz novi dizajn Borland je morao razviti čitavu biblioteku rutina za dijaloge i maske za unos. Ta biblioteka i prateće rutine zauzimaju ne baš zanemarljivo mesto na disku, i onako opterećenom veličinom paketa. Imamo utisak da Borland pokušava da stvori neki novi vizuelni standard, pa čak nagovara i nas (programere) da ga prihvatimo. U tom slučaju moramo da koristimo čitavu seriju poziva i tehnika različitih od onih standardnih. Da li je potrebno da pored svih poziva rutina koje nam nudi standardni *Windows* učimo i koristimo alternativne? Pa, *Windows* i jeste napravljen zato da bi SVI programi koristili iste biblioteke i smanjili zauzeće prostora na disku. Šta bi se desilo kada bi svi novi programi uvodili svoja rešenje u *Windows*? Mislimo da je ovo izmišljanje tople vode Borlandu bilo nepotrebno, ipak, spremni smo da tu zamerku zaboravimo pred ostalim mogućnostima paketa. Kao što je bilo rečeno pre nekoliko brojeva: „Ima ljudi kojima se i to sviđa".

Kada se instalira, *Turbo C++* formira posebnu grupu u okviru *Windows*-a. U njoj se osim samog C++ kompajlera nalazi i nekoliko pomoćnih programa: program za uvoz DLL *Windows* biblioteke, program za konvertovanje teksta iz ANSI u OEM standard i obratno, *Workshop* o kome ćemo reći više malo kasnije i *Turbo Debugger* for *Windows*, *Borland*-ov debager za *Windows* programa, dosta kontraverzno izveden.

Svi pomoćni programi (osim debagera) su korektno napisani u *Borland Windows look* stilu i nemamo zamerku na njihov rad – jednostavno vrše svoj posao.

KAKO ISTERATI BUBICE?

Već je poznato da *Borland* uz svoje verzije kompajlera koji rade pod *Windows*-om daje debager kao poseban program. Ovaj program je napisan drugačije od svih koje smo do sada videli, naime, svi *Windows* programi se pridržavaju jednog izgleda i načina rada – nema divljeg preklapanja grafičke i tekst video memorije – znači svi rade u grafičkom modu. Nije nam sasvim jasno kakve su probleme susreli *Borlandovi* programeri pri pisanju ovog programa, ali se ima utisak da se njegovim izvršavanjem krši dosta pravila za korektan rad. Daleko od toga da je rad nepouzdan – samo je čudan.

Kada startujete *Turbo Debugger for Windows* naćićete se u tekst-modu, a program koga debagujete će i dalje raditi pod *Windows*-ima u grafičkom modu. Ukoliko imate dva ekrana i dve grafičke karte na jednom računaru (EGA/VGA i Herkules) onda je utisak sasvim povoljan. Ovaj program ima *dual screen* opciju pomoću koje, na primer, *Windows* redi na VGA a debager na Herkulesu, pri čemu ne smetaju jedan drugom i sve teče glatko. Ako nemate dva ekrana morate biti spremni na neprijatno preklapanje tekst i grafičkog ekrana, u zavisnosti od toga šta kad treba da se prikaže. Daleko bolje bi bilo da je Borland uz ovaj kompajler dao (možda kao dodatak uz ovaj debager) i debager koji radi legalno pod *Windows*-ima. Čak i debager sa manje mogućnosti, pa i bez onih osobina *Turbo Debager*-a koje se ne mogu podržati pod *Windows*-ima, bi bio primereniji za svakodnevni rad. Inače, ako zanemarimo tekstualno/grafički rad ovog debagera, to je sasvim dobar proizvod, sposoban za najteža ispitivanja

Listing 1

```
// Primer koji pokazuje lakocu rada sa object windowsom
// Kompletan editor klase notepada u nekoliko linija !!!

#include "owl.h"
#include "filewnd.h"

// Deklarisanje TFileApp
class TFileApp : public TApplication {
public:
        TFileApp(LPSTR AName, HANDLE hInstance, HANDLE hPrevInstance,
                LPSTR lpCmdLine, int nCmdShow)
                : TApplication(AName, hInstance, hPrevInstance, lpCmdLine,
nCmdShow) {},
        virtual void InitMainWindow();
        virtual void InitInstance();
};

// Deklarisanje TMyFileWindow
class TMyFileWindow : public TFileWindow {
public:
        TMyFileWindow(PTWindowsObject, LPSTR, LPSTR);
};

// Ucitavanje menija
TMyFileWindow::TMyFileWindow(PTWindowsObject AParent, LPSTR ATitle,
  LPSTR AFileName)
  : TFileWindow(AParent, ATitle, AFileName)
{
  AssignMenu("FileCommands");
}

// Start glavnog prozora aplikacije klase TMyFileWindow
void TFileApp::InitMainWindow()
{
  MainWindow = new TMyFileWindow(NULL, "Editor", "");
}

// Ucitavanje akcelerator tabele (hot keys)
void TFileApp::InitInstance()
{
  TApplication::InitInstance();
  if ( Status == 0 )
        HAccTable = LoadAccelerators(hInstance, "FileCommands");
}

// Start aplikacije
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
  LPSTR lpCmdLine, int nCmdShow)
{
        TFileApp FileApp ("FileApp", hInstance, hPrevInstance,
                lpCmdLine, nCmdShow);
        FileApp.Run();
        return FileApp.Status;
}
```

Editor u samo 50 programskih redova

programa. To i nije tako čudno, ako se zna da je ovaj debager nastao iz standardnog *Borland* debagera, koji sa razvija već duže vreme, uz razne verzije paskala i C-a za DOS dopunjen mogućnostima praćenja poruka koje su glavni način za komunikaciju između aplikacija u *Windows*-u i drugim *Windows* specifičnostima.

LETIMIČNO PO SORSU

Kao poseban deo ovog prikaza, izdvojili smo jedan deo C kompajlera – *browse* opciju (eng. *browse* – letimično pregledati)

Kada programer prvi put počinje da koristi objekte i da objektno programira, nailazi na probleme. Jer, i ako je sve jasno, nedostaju neka osnovna objašnjenja o osobinama objekata i njihovom korišćenju. Sa druge strane, konačno postaje nemoguće ignorisati objektno programiranje, jer se bez njega ne mogu iskoristiti sve one lepe stvari koje nudi *ObjectWindows*. Autor ovog teksta se našao u sličnoj situaciji (do sada uspešno eskivirajući OOP) kada je trebalo preći na *Windows* programiranje, i to ne samo u C-u nego i u jeziku *Turbo Pascal for Windows* i u *Visual Basic*-u.

Browse opciju možete naći u glavnom meniju kompajlera. Ona omogućava pregled „povezanosti" nekog dela programa sa drugim delovima, pregled objekata koji su iskorišćeni, pregled funkcija i promenljivih – ukratko, sve što vas zanima. Kako je to realizovano, posebna je priča.

Da prvo vidimo kada je ova opcija korisna. Pa, na primer, kada je potrebno izmeniti deo ranije napisanog programa (pisanog, recimo, pre nekoliko meseci – dovoljno vremena da se zaborave neke „veze" i zavisnosti u programu) koji treba u kratkom roku izmeniti i prilagoditi novim zahtevima. Tada se jednostavno startuje **browse**, pogledaju se procedure i promenljive, provere pojavljivanja i zavisnosti objekata u programu i vidi (po modulima) šta od čega zavisi. Da li vam se nekad desilo da izmenite deo programa koji je „apsolutno siguran" a da vam program padne na drugom mestu jer ste u modulu X zaboravili promenljivu Y? Verujte – raziv osećaj, ali ne više i obavezan – **browse** sistem isključuje mogućnost takve greške. Ili, što je još važnije, kako izmeniti tuđ program, program koji je „firma platila budžašto" a koji mora da dobije tu i tu opciju? Operacija je jednostavna ako imate **browse**.

Nemamo dovoljno reči hvale za ovu opciju, posebno zbog toga što je do maksimuma iskorišćena *Windows* okolina – pogledajte samo sliku na kojoj se vidi povezanost objekata koja prati primer uz ovaj tekst. I laicima je jasno koji objekt ima „naslednike" a koji „potomke". Ako se „klikne" mišem na bilo koji objekat sa slike, dobija se precizniji spisak metoda za korišćenje tog objekta, njegovi naslednici i preci, a vidi se i pojavljivanje tog objekta u listingu.

Ovo je, konačno, alat koji približava OOP i laicima, prikazujući grafički ono o čemu se do sada samo pričalo.

NEOGRANIČENI RESURSI

Jedan od odvojenih pomoćnih programa koji se izdvaja svojim kvalitetom je *Resource Workshop* – program za dizajn resursa.

Pre pojave *Windows*-a, pri programiranju u DOS-u, svaki programer je pravio svoja programa tako što bi isprogramirao menije i maske za unos po svom ukusu, u jeziku u kome je radio ostatak programa, pa sve to kompajlirao u konačan program. *Windows* je stvarno napravio revoluciju u svemu, pa i u programiranju, tako da sada program koji radi pod *Windows*-ima nema programski određen korisnički interfejs već se sve maske za unos, meniji, ikone, dijalozi i ostali grafički elementi programa nazivaju resursima i uključuju u program na način različit od DOS-a. Tako smo dobili resurs editore, programe koji su u stanju da na jednostavan grafički način pripreme resurse potrebne za naš program. Do sada smo se sreli sa resurs editorom koji je išao uz *Borland* C++ 2.0 – *Borland* ga je otkupio od drugog proizvođača u vreme kada nije mogao da ponudi svoj proizvod. Od tada je *Borland* imao dosta vremena da osmisli i napravi editor resursa dostojan firme koja ga prodaje.

Pristup je donekle različit od editora na koji smo navikli, ali to nam nije smetalo da se odmah naviknemo na novi. Rad je veoma dobar i „sve opcije su na svome mestu". Dovoljne bi bile i dosad nabrojane osobine da ga trajno prihvatimo. Ipak, nije to ono što nas je oduševilo. Ovaj editor ima neverovatno širok spektar resurse koje podržava: od običnih RC (resurs skript fajl) i RSC (resurs objektni fajl) fajlova (koji su standardno bili uključeni i u stari editor) preko standardnih ICO (Windows ikona), BMP (slika), DLG (dijalog), CUR (kursor), EXE, DLL fajlova, do egzotičnih stvari kao što su FON (*Windows* font), FNT (*Windows* biblioteka fontova) i DRV (*Windows* drajver). Da, sada je moguće ne samo dodavati naša slova u *Windows* po želji, nego i editovati *Windows* drajvere.

Recimo, u *Windows* drajveru za ekran (hercules.drv) se nalazi nekoliko fontova, bit-mapa i kursora. Ovo nas je izazvalo da za samo nekoliko minuta potpuno promenimo izgled svih *Windows* elemenata – strelica na skrol površinama, kursora.

Drugi plus je što on ne samo da čita već i snima ta formata podataka, tako da smo iz COMMDLG.DLL (*Windows* biblioteka koja sadrži neke standardne dijaloge – za izbor fonta, za snimanje fajla itd.) uspeli da „ukrademo" dijalog za izbor fajla koji se učitava i da ga snimimo kao RC – dakle, u potpuno čitljivom skript obliku!

Eh, sa ovakvim programima, zabavi nikad kraja.

U SKLADU S TRADICIJOM

Ko se već bavio Borland-ovim C kompajlerima zna da je njihov kod dosta pouzdan i dosta brz. Na stranu to što su maksimalno optimizovan i neoptimizovan kod slični po dužini (Microsoft C tu pravi drastičnu razliku), ali ne i po brzini.

Ni ovaj kompajler se ne razlikuje od ostalih iz firme – uostalom, i on je samo „istrgnut" iz *Borland C++ 3.0* paketa. Ovo se potvrđuje i time što u jednom od menija C-a imamo opciju za podešavanje editora. Stvar je u tome što je postavljena i opcija za DOS verziju, a nje u ovom paketu jednostavno nema.

Problem testiranja kompajlera za *Windows* okruženje postaje sve izraženiji: kako testirati program koji radi u multitasking okruženju i njegova brzina zavisi ne samo od kvaliteta koda nego i od zauzeća procesora? Teši nas činjenica da je „kod za DOS bio kvalitetan, pa ovaj ne može da bude lošiji."

Optimizacije i načini za njihovo izvođenje su još uvek bolna tačka svih PC kompajlera – velika optimizacija stvara potencijalno nestabilan kod. Do sada nam se nije desilo da program rađen nekim od Borland kompajlera neće da radi zbog optimizacije. Ovo naglašavamo zbog toga što je o optimizaciji Borlandovih kompajlera bilo malo reči na stranicama *Računara*. Naše je mišljenje (a neki manji testovi su to dokazali) da kod koji optimizuje *Borland C* nije za potcenjivanje.

ŠTA NEDOSTAJE?

Ako Borland prodaje ovaj kompajler kao „manje profesionalan", koje od alatki smatra nepotrebnim u amaterskom radu, odnosno šta je izbacio iz ponude u odnosu na *Borland C++ 3.0*?

Na prvom mestu, ono što upada u oči je nedostatak Turbo asemblera – odvojenog asemblera, TASM-kompatibilnog i jako fleksibilnog. Turbo C ima mogućnost generisanja koda u sred C-a uz pomoć ASM direktive (osobine koju ima još od BC++2.0 verzije), koja dozvoljava mešanje C-a i asemblera po želji: malo C, pa malo ASM, pa opet C.

Kada smo malo razmislili, zaključili smo da TASM tu i nema šta da traži – komandne linije iz koje se on startuje nema, pa nema ni elegantnog načina da se pokrene. A pisanje *Windows* okoline samo za TASM bi ipak bilo preterivanje.

Druga stvar je *Turbo Profiler*, u teoriji jako koristan program. Profiler je program koji analizira generisani kod programa i brzinski ispituje kritična mesta (petlje i brojače), koja mogu da utiču na brzinu programa. Kada sve ispita, profajler daje izveštaj o tome, tako da programer može da poveća brzinu kritičnu za neke aplikacije. Takođe, stvari kao što su višak koda ili nepotreban kod u okviru petlje se lako detektuju. Savršen alat, ako vam treba savršeno ispeglan kod, maksimalno brz. Kao što rekosmo, ako vam aplikacije nisu brzinski kritične i ako nemate višak vremena za igranje sa brzinom, nedostatak profajlera i nije neki gubitak. Na primer, u dosadašnjem radu (uz BC++2.0) smo profajler startovali par puta, više probe radi nego zbog stvarne potrebe.

Time se scrpljuje spisak nedostataka.

IMATI ILI NE?

Ovaj paket je veoma dobro izbalansiran i ne verujem da mu se može naći neka veća zamerka. Uz svog brata, *Borland C++ 3.0*, ovaj paket danas čini vrh ponude na tržištu. Ako vam pri tome nije važno programiranje pod DOS-om, nije na odmet uštedeti koji deo osne većeg paketa.

Naravno, treba videti i odgovor Microsoft-a na Borlandove izazove, mada uz mogućnost *upgrade*-a (prelaska na novu verziju uz doplatu u odnosu na staru) teško da možete biti na gubitku.

Odluka „imati ili ne" u vreme gotovo izjednačene trke Microsoft-a i Borlanda se svodi na pitanje ličnog ukusa – niti će jedan proizvođač mnogo isprednjačiti po osobinama ponude niti će drugi mnogo stagnirati - jednostavno, u jednom trenutku vodi jedan, u drugom drugi. Programeri u uvek ne dobitku.

Korisna adresa:

CET
11000 Beograd, Skadarska 45
Tel. 343-043



PARADOKS ZA MASE

Borland je jedna od retkih firmi koja se odvažila da u vreme kada ja dBASE već pustio duboke korene iansira sosptveni standard za baza podataka. Iz činjenice da upravo ovih dana na tržište izlazi četvrta generacija programa *Paradox* aasvim se mirno može izvući zaključak da se Borland i na ovom polju iapo anašao.

Dušan Popović

U PC svetu postoji nekoliko standerde za baze podataka. Pošto su dBASE II i dBASE III napravili bum u radu sa bazema podeteka (jeste bilo poodavno, ali ko prvi standardu – njegov standard) i pošto je dBASE tormet postao gotovo uslov i standard prenosivosti podataka i eplikacija, bile je popriična hrebrost početi nešto novo. Borland, kao dosta jaka firma, sposobna da nameće standerde, ipak definiše sopstveni standard koji dalje konsti u svojim programima.

Već smo pisali o prenosivosti programa između različitih Borlend paketa. Pri tome smo spomenuli da između oslalog, sa *Paradox* formetom podateke radi i *Paradox* interaktivni paket za baze podataka – tada u verziji 3.5, a sada 4.0. Pošto je reč o „standardu različitom od standarda", prvo da vidimo kako je Borland organizovao svoju bazu podataka.



IZGLED

Red je (za nekoga ko se navikao na *Borlend* pakete) sasvim prirodan – tu su svi elementi na koje smo navikli iz poslednjih verzija Borlendovih jezika. Imamo meni na vrhu ekrana, dijaloge, skrol barove, preklapajuće prozore itd. Ono što ćete odmah primetiti je nedostatak (ili promena standarda) *hot-keys* kombinacije na koje smo navikli. Ne radi Alt+broj prozora, Alt+F3, pa čak ni Alt+slovo iz menija. Do menija sa dolazi pritiskom na F10, pa tek onda korišćenjem strelica ili istaknutog slova iz menija. Da bi zatvorili aktivan prozor morate ići preko kombinacije Alt+razmaknice, pa onda C, i tako dalje. Ukratko, imaćete malo muke da se naviknete. Ono što u stvari zbuni čoveka je to što uz *Borland look* očekujete i *Borland hot-keys*, a toga neme. U radu vam je gotovo neophodan miš. To kažemo zbog toga što još uvek važe sve operacije sa prozorima, menijima i mišem. Smatramo da je lakše kliknuti na strelicu nagore u gornjem desnom uglu svakog prozora (povećanje prozora na ceo ekran), nego prozor povećavati pritiscima na Alt+razmaknica pa onda M.

TABLICE

Reč je o poznatom pojmu: bazu podataka možemo zamisliti i kao tablicu sa **n** polja puta m zapisa. Zapisi mogu sadržavati različite vrste podataka – stringove, numerička polje, datume, memo polja kao i BLOB polje. BLOB polja su izuzetno moćna jer mogu da sadrže binarne podatke ogromne veličine. Nije teško napraviti bazu podataka gde je jedno od polja i BLOB koje sadrži sliku u pcx formatu. Na primer, baza koja sadrži celokupnu ponudu neke robne kuće ili baza delova lica (kako li je Interpol radio bez *Paradox* a?)

Definisanjem tablice (ili učitavanjem već urađene) počinjemo rad sa nekom bazom podataka. Tablicu je moguće definisati na više načina: moguće je početi iz početka, ali ako u nekoj tablici već postoji struktura podataka koja nam odgovara – iskoristićemo je. Tablica se definiše tako što se u posebnom prozoru u polja tablice na ekranu unose vrednosti i imena. Na primer, pod 1 unesete kao ime „Broj lične karte" (ili bilo koji drugi naziv), a pod tipom polja unesete „N" i dobili ste prvo numeričko polje.

Tablica je u *Paradox*-u osnovni činilac baze podataka. Tablicu možete pregledati, menjati, štampati kao izveštaj ili vršiti neku od mnogobrojih operacija sa njom.

Ono što ćete verovatno prvo pokušati je opcije **View** u glavnom meniju. Ona nam omogućava da tablicu pogledamo ili menjamo. Tu dolazimo do jedne zamerke. Kada pozovete **View**, i, uglavnom, kada zahtevate operacije sa diskom, pred vama se pojavi prozor za dijalog čije rešenje naprosto smeta ovako dobrom paketu. Odmah vam je ponuđena mogućnost da unesete ime tablice ako ga znate 999999t ili da pritisnete Enter, pa da vam *Paradox* ponudi spisak tablica. Sve je to lepo, ali ne postoji opcije za promenu radnog direktorijuma i diska! A šta ako se tablica sa kojom želite da radite ne nalazi u tom direktorijumu? Ništa vam ne preostaje nego da izađete u glavni meni i da preko ko_zna_koje opcije menija *Tools* pređete u drugi direktorijum. Sa druge strane, ako pogledamo kom segmentu tržišta je namenjen ovaj paket, i ako se setimo kako sekretarice i polupismeni operateri znaju da zalude čak i neki savršeno smišljen program – možda je i bolje što je opcija malo zabačena.

Tablicu možete i menjati – pritiskom na F9 prelazite u takozvani *edit mod* gde jednostavno menjate postojeće i dodajete nove podatke u tablicu. Pošto je rad u mreži gotovo neizbežan kod ovakvih programa, postoji i opcije *coedit* koje se poziva sa Alt+F9 i koja omogućava korisnicima mreže, ne×terminala, da istovremeno menjaju bazu podataka. Ako je za slog podataka sa kojima trenutno radite potrebno zabraniti pristup drugim korisnicima baze, iskoristićete zaključavanje sloga (Alt+L) i mirno ćete ga menjati bez upitanje drugih korisnika.

O bazi možete postavljati i „pitanja", odnosno možete vršiti operacije sa podacima iz baze, kao na primer: sumiranje, prosek i još mnogo toga.

Tablicu možete sortirati i indeksirati – deo oko indeksiranja (a podržava primarne i sekundarne indekse) smo već obradili tokom prikaza *engine*-a. Sekundarni indeksi su jedan od specijaliteta *Borlanda*. Tablicu je moguće otvoriti više puta po istom indeksu, više puta po različitom, ili više tablica po više indeksa – sve jednostavno i pregledno na ekranu, a sortirano na sve moguće i nemoguće načine.

Postoji i lepši način za pristup bazi podataka od tabličnog pregleda – preko formulara. Pregled podataka uz pomoć formulara izgleda ozbiljnije, a omogućeno je i dosta dodatnih pogodnosti.

FORMULARI

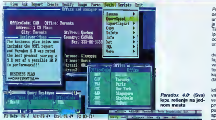
Formulari služe za lepši i bolji pregled i rad sa određenom tablicom. Formular možemo sami definisati, a možemo iskoristiti i neki od već postojećih. Pri definisanju formulara se nalazimo u posebnom prozoru gde dodajemo elementa po želji – okvire i podvlačenja, sve u cilju veće preglednosti podataka. Podatke iz baze raspoređujemo po volji, grupišemo ih tako da idu logičnim rasporedom. Formular može da bude veći od ekrana (po vertikali), može da sadrži samo delove neke baze podataka, ali može da istovremeno sadrži i više baza. Ovo je jedno od značajnih oružja za pravljenje aplikacija.

U formular ulazimo pritiskom na F7, kada se već nalazimo u tabelarnom pregledu. Izmena podataka teče normalno, ako se naviknete na neka pravila. Prelazak na sledeću stavku za unos se vrši TAB tasterom. Program sam vrši kontrolu pravilnosti vrednosti koje unosite, tako da ne možete uneti slovo u numeričko polje i slično. Tu imamo još jednu zamerku: pošto program podržava više standarda unosa datuma (u datum polja) da bismo uneli neki datum moramo da otkucamo dan (recimo 02), a onda i tačku da bismo nastavili sa unosom meseca. Mislimo da je unos mogao malo inteligentnije da „sam" odredi gde ide separator.

Formulari su veoma korisni kada razvijamo sopstvene aplikacije – otvorimo bazu, definišemo je, definišemo formular i uz još neznatan posao imamo kompletnu aplikaciju.

PITANJA I ODGOVORI

Kada se pravi baza podataka, ona ne služi samo da čuva podatke, nego ih je po pravilu, potrebno kasnije obraditi, izdvojiti potrebne, grafički ih prezentovati i slično. Sve to (osim grafika) je dostupno preko opcije **Ask** glavnog menija. Kada pozovemo **Ask** dobijemo (da,



Paradox 4.0: (Sva) lepa rešenja na jednom mestu

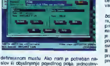
opet onakav) dijalog za izbor imena baze. Možemo pozvati bazu sa kojom radimo ili neku novu, a u svakom slučaju se otvara poseban prozor sa malo izmenjenim tabličnim pregledom. Sada možemo krećući se strelicama da u pojedina polja tablica postavljamo uslove koji nas zanimaju. Spisak uslova se može dobiti preko helpa (F1), a pošto nismo imali pisanu dokumentaciju, dobro nam je poslužila i takva pomoć. Neke uobičajene uslove možemo udariti i uz pomoć funkcijskih tastera. Pritiskom na F6 polje na kome se nalazi kursor postaje čekirano, na njemu je ispunjen uslov pri pretraživanju, pa se pojavljuje u prozoru **Answer** u kome *Paradox* izlistava sva polja koja odgovaraju uslovu.

Često nam baza podataka služi da u njoj čuvamo velika količina numeričkih polja. Tada je korisno takvu bazu prikazati u obliku grafika, ali bez komplikovanih izvoženja podataka u grafičke programe. Zbog toga je program obogaćen opcijom za prikaz grafika. Postoji dosta veliki broj mogućih prikaza (2d i 3d, linije, torte i površine) koji nam omogućavaju da, ako ne da napravimo reprezentativna grafika, bar sami vidimo kakvim ciframa baratamo. Postoji i pojam *default* grafika, u stvari, sva podešavanja koja uradite se pamte, tako da ćete (kada sledeći put upotrebite **graph**) raditi sa vašim omiljenim grafikonima. Naravno, ove opcije može da štampa u fajl, na ekran ili štampač, tako da su podržani svi uređaji. Ukoliko imate neku od egzotičnih grafičkih kartica, smatramo da nemate razloga za bogu tipa „da li ću na ovoj grafičkoj karti moći da vidim grafik?", jer *Paradox* radi sa grafikom uz pomoć, zamislite, standardnih *BGI drajvera. Mislimo da ne postoji video karta za koju ne postoji ovakav drajver.

IZVEŠTAJI

Nekada je potrebno izdvojiti određene zapise iz baze i prikazati ih na ekranu ili štampaču. Izveštaj se pravi preko **report** opcije glavnog menija. Izgled izveštaja je moguće definisati u zavisnosti od potreba, izgled menija će, naravno, zavisiti i od mogućnosti vašeg štampača – stvari kao broj slova u redu i margine su podrazumevane, ali nije na odmet mogućnost dodatnog podešavanja.

Zanimljiv je način na koji se raspoređuju polja po fiktivnom „papiru" koga vidimo na ekranu. Moguće je, na primer, staviti datum polje u gornji desni ugao ekrana i pri štampanju na svakom novom papiru imamo datum na



definisanom mestu. Ako nam je potreban naslov ili objašnjenje pojedinog polja, jednostavno ga dopišemo. Izveštaji se (kao i formulari) snimaju u „bazu izveštaja", što nam kasnije omogućava da jednostavno izaberemo već formiran izgled izveštaja. Nije zanemarljiva činjenica da jedna firma radi i sa nekoliko desetina standardnih izgleda izveštaja. Kada jednom definišemo sve potrebne izglede, dalje ih bez razmišljanja koristimo. Ovo je potencijalno još moćnije ako zamislimo veliko preduzeće koje pod mrežom radi u *Paradox*-u. Tada se svi standardni izveštaji nalaze na jednom mestu, a svi ih koriste po želji.

RAD U MREŽI

Osnovno tržište kome je ovaj program namenjen su velike firme, one koje obrađuju velika količina podataka i one u kojima dolazi do velikog protoka podataka (aerodromi i hoteli?). Spisak mreža koje su podržane je impresivan, ali se nećemo ponavljati.

Pošto se o radu pod mrežom dosta razmišljalo, u sama korena sistema su ugrađene ideje za lakav rad. Lak i logičan rad pokazuje da program nije samo jedan od krpljenih sistema za baze podataka. Već smo spomenuli rad u *coedit* modu i zaključavanje slogova baze. Ovakav rad smo isprobali na *Novell* mreži sa neposvećenim serverom i šest računara. Dosta vremena smo potrošili „igrajući" se ovim programom i nismo uspeli da ga dovedemo u nestabilno stanje.

Kada se radi pod mrežom, postoji potencijalno veoma velika grupa drugih korisnika koji baš ne bi trebalo da zaviruju u vašu bazu podataka. Zato se svaka baza može staviti pod šifru. Nju znate samo vi i možete biti sigurni da vašu bazu niko neće videti bez odobrenja.

PAL

Stigli smo i do programiranja. PAL je *Paradox Aplication Language* – jezik za pravljenje *Paradox* aplikacija. Vidi se da *Paradox* sledi osnovnu ideju *dBASE* paketa: sve je moguće uraditi iz menija, ali za naprednije korisnike tu je i mogućnost pravljenja programa i aplikacija. Time se poslovi standardizuju i ubrzavaju. Ipak, *Paradox* je ovu ideju podosta razradio u odnosu na uzor, tako da je neke stvari moguće uraditi na nekoliko načina.

Najlakša stvar koju je moguće isprobati je definisanje makroa. Pošto se i makro snima u fajl u obliku koji može da se edituje i dopunjava, možemo da imamo program (u *Paradox* terminologiji *script*, sa ekstenzijom .sc) koji je kombinacija *Paradox* komandi i pristupa standardnim komandama u meniju.

PAL je jezik lak za razumevanje, ne sadrži kriptične skraćenice, tako da nakon malo ispitivanja demo programa (koji idu čak i uz beta verziju *Paradox*-a) možete početi sa pisanjem prvih skript programa. Naravno, da biste potpuno razumeli PAL komande i pisali velike aplikacije, potrebna vam je dokumentacija koju mi nismo imali. Uzimajući u obzir ranija iskustva sa Borlandovim paketima, očekujemo da će uputstva koja će ići uz *Paradox* biti razumljiva i lepo napisana.

Iz PAL-a imate mogućnost kontolisanja *borland look* menija, funkcijskih tastera, prozora, dijaloga, formulara i slično. Moguće je napraviti aplikaciju koja deluje veoma profesionalno. Postoje neki standardni postupci na koje se čovek navikne u radu sa *Paradox*-om (na primer, postojanje opcija Do-it i Cancel u skoro svakom meniju, gde one znače vraćanje nazad ili potvrđivanje), pa nema razloga da se pri pravljenju naše aplikacije ne pridržavamo tih pravila.

Ipak, postoji i efikasniji način za pisanje *Paradox* programa od samog pisanja – programe je moguće gotovo crtati uz pomoć *Workshop*-a.

WORKSHOP

Ovaj deo paketa se gotovo skriva u „sistemskom" delu menija (krajnje leve opcije glavnog menija). Toliko nas je obradovalo postojanje takvog alata, da smo gotovo polovinu vremena testirali samo njegove mogućnosti.

Borlandu je prešlo u naviku da svakom svom jeziku dodaje i *Workshop*. Naravno, ta radionica se od jezika do jezika razlikuje, ali joj je glavna osobina da drastično olakšava rad. Radionica koja ide uz *Paradox* je potpuni sistem za dizajniranje aplikacija. Pri tome se preko menija definišu meni koji koristi vaš program, podmeniji, dijalog-boksovi, prozori i akcije koje se aktiviraju određenom stavkom menija. Najbolja stvar u svemu tome je što se svi standardni delovi aplikacije (dijalog-boks, na primer) mogu predefinisati. Tako dobijamo proizvoljne tastere na dijalogu, proizvoljan tekst, i sami dodajemo akcije koje se mogu aktivirati.

Uz svaki deo aplikacije se može dodati help. Time se dobija neverovatno dobro rešen help sistem. Može se ići do detalja kao što su help za dijalog-boks ili za neku opciju menija. Do sada nismo videli lakši način za definisanje ovog važnog detalja. Pošto je u drugim programskim jezicima dosta komplikovano dodavati poziciono osetljiv help sistem, od toga se uglavnom odustaje. Uz ovakvo rešenje, prosto je šteta ne napraviti ga.

Opcijama menija mogu se, kao što smo rekli, dodati podopcije (*pull down* meni) ili akcija. Pod akcijom se podrazumevaju različite stvari od PAL naredbi do izvršenja drugih skriptova.

Da bi zadovoljstvo bilo potpuno, *Workshop* omogućava generisanje dokumentacije (!) za upravo napravljeni program. Granenje programa ili menija se može i odštampati, tako da imate potpun podsetnik o tome kako radi vaša najnovija kreacija.

Kada ste zadovoljni svojim programom, možete generisati izvršni kod. Ovo „izvršni" se može smatrati uslovno, dok nam do ruku ne dođe obećani *runtime* modul. Program će, istina, generisati potreban skript i prekopirati potrebne biblioteke u poseban direktorijum, ali taj skript može da se učita u bilo koji editor, a samim tim i da se menja. To znači da možete praviti programe za svoje potrebe, ali da je pravljenje programa za tržište gotovo nemoguće. Da bi programi mogli da se slobodno prodaju, moraju biti kompajlirani (ili bar kriptovani) i nezavisni od (ipak velikog) *Paradox* paketa. Sve to će biti ispunjeno pojavom *runtime* modula u koji polažemo velike nade, ukoliko uspe da ispuni osnovne uslove: treba da bude mali i brz.

UVOZ – IZVOZ

Ono što nas je brinulo kod ovakvog paketa je pitanje: šta raditi sa postojećim podacima koje koristimo, koji su u *dBASE* formatu i kojih nije malo? Borland nije zaboravio ko je stariji, pa omogućava uvoz i izvoz iz dosta velikog (i sasvim dovoljnog) skupa formata podataka.

Moguće je uvesti i izvesti podatke u formate kao što su *dBASE* ili *123* - da pomenemo samo najvažnije. Verujemo da su i ova dva vrste formata dovoljne za prenos podataka „iz svega i u sve", a da i ne govorimo o mogućnosti da je vaša bivša(?) omiljena baza podataka direktno podržana.

Kompatibilnost sa *dBASE* formatom smatramo definitivnom, jer kada je program uspeo da „proguta" nekoliko naših baza sa velikim memo poljima i da ih konvertuje u *Paradox* format, onda nema problema ni sa ostalim „manjim" i „lakšim" bazama.

MOGUĆE PRIMENE

Čemu ovakav paket teži nije teško pretpostaviti. Iako je sasvim pogodna i primena za jednog korisnika, prava moć paketa se vidi u velikoj mreži računara. Na takvo tržište ukazuju i neka rešenja koja smo videli. Na primer, program čak zahteva da mu se ne unosi ekstenzija pri pregledu fajlova na disku. Zašto? Pa zato što ljudi koji treba da rade sa ovim programom ne treba da znaju šta je ekstenzija. Ne treba da znaju ni šta je direktorijum, pa eto odgovora zašto se direktorijumi menjaju iz „dole levo levo" menija.

Sjajan primer kalibra firme koja koristi *Paradox* je Koka-kola. Čujemo da cela firma radi uz pomoć *Paradox*-a, a zar je potrebno govoriti o veličini Koka-kole?

Naravno, ni jednokorisnički sistem nije hendikepiran – razlog zbog koga toliko potenciramo rad u mreži je taj da do sada nismo videli paket koji tako lako i jednostavno podržava mreže.

UTISCI

U opasnosti smo da preteramo sa pohvalama, ali retko se nalazi toliko lepih rešenja na jednom mestu. Paket koji smo testirali je beta verzija koja naravno dolazi bez dokumentacije i sa potencijalnim bagovima. Ono što smo mi videli je jako dobar, pouzdan i brz paket. Sa beta (gama?) verzijom ima veze utoliko što osim već pomenutog nedostatka knjige ima i „konspirativno" ime koje se na nekoliko mesta provlači kroz doc fajlove: „diamond gamma". O bagovima nema ni reči (ni uz najbolju volju nismo uspeli da destabilizujemo program) i imamo utisak da smo dobili verziju koja „samo što nije objavljena".

Ovim programom je krug definitivno zatvoren: *Paradox Engine – Paradox 4.0 – Object Vision*. A autor je, kao nepopravljivi ljubitelj Borlandovih kreacija, uspeo da još nekoliko paketa drugih firmi (*Clipper* i *dBASE*) izbaci iz upotrebe. Čak i kada je potrebno bazu podataka dati nekome ko koristi neki stari (zastareli?) paket? Ništa strašno, baza se samo konvertuje, a i dalje se radi u *Paradox*-u.

PARADOX FOR WINDOWS

PARADOX ZA KLASE

***Paradox for Windows* smo dobili kao beta verziju, koju *Borland* naziva *amber beta*. Sa velikim nestrpljenjem smo lepitali ovaj paket, znajući kakve sve mogućnosti dobija jedna baza podataka kada se prenese u grafičko okruženje kakvo je *Windows*. Da se u očekivanjima nismo prevarili, pokazalo se već pri pregledu demo baza koje su priložene uz paket.**

Da se radi o paketu za posebne (gornje) klase korisnika biće vam jasno kada pogledate zahteve: oko devet megabajta na hard disku (što je dva puta više nego kod DOS verzije) i minimum četiri megabajta memorije (!). Kod ovog drugog podatka smo od neverice trljali oči, ali Borland je neumoljiv – da bi program uopšte mogao da radi, treba vam četiri megabajta, što znači da vam je za iole ozbiljniji rad (da bi se izbeglo preterano korišćenje hard diska) potrebno bar šest do osam megabajta RAM-a. Prosto je neverovatna progresija kojom noviji programi povećavaju zahteve za resursima računara – dok je nekad bilo „i previše" imati jedan megabajt, sada ni četiri nije dovoljno.

Uz program je priložen i *readme.doc* u kome su upisane (a ima ih bar tridesetak) greška i nedostaci koje Borland namerava da popravi pre puštanja ovog paketa u prodaju. Drugim rečima, ovaj paket (za razliku od *Paradox 4.0* koji je, koliko je nama poznato, bez gešaka) treba još puno da se popravlje do konačne verzije.

Da bi se počelo sa radom, potrebno vam je devet megabajta na hard disku i oko dvadeset minuta vremena. Ukoliko ste upoznati sa radom na DOS verzijama *Paradox*-a, tada nema problema – sve mogućnosti su tu (možda poneka na drugom mestu u meniju) i sve je „kao u DOS-u". Dodat je **speed** bar (linija sa ikonama) i ostali specijaliteti Borlanda. Uvođenje sopstvenog standarda u korisnički interfejs *Windows*-a smo već kritikovali, tako da samo konstatujemo da je sve tu – Borlandovi dijalozi, ikone, pa čak i predefinisani skrol barovi, a mesta na disku sve manje.

Razliku ćete primetiti pre svega u lakoći kretanja kroz menije – sada se za to brine *Windows*, tako da kombinacije Alt+taster (koje rekosmo da zaboravite u DOS verziji) ovde rade. Tu je i (konačno civilizovan, jer je kretanje po disku sasvim lako) **file** selektor koji u odnosu na DOS verziju ima još neke prednosti. Kada zatražite otvaranje baze podataka, dobijete spisak svih .db i .dbf fajlova. Db je *Paradox* baza, ali dbf je standardni dBASE fajl.

Ovo znači da fajl selektor izlista sve što liči na fajl baze podataka, a ako nije u Paradox formatu, on to prepozna i konvertuje ga. Sjajna mogućnost, ali nam nije jasno zašto je nema u DOS verziji, nego je potrebno bazu konvertovati preko posebne stavke menija.

Rad sa podacima je sličan radu u *Paradox*-u 4.0. Tablicu je moguće pregledati u normalnom ili u „form" izgledu. Pošto je *Windows* grafičko okruženje, izgled formulara pod njim izgleda i te kako bolje u odnosu na DOS. Dok smo u DOS-u imali solidnu kontrolu izgleda formulara, ovde je ona zaista izvanredna. Formular može da prikaže BLOB (*Binary Large OBject*) polje kao sliku. Tu do izražaja dolaze sjajni demo-fajlovi, u kojima imamo pregled vrsta riba (sa slikama svake od njih) ili aviona. Takođe do izražaja dolazi i brzina (sporost) rada računara sa objektima koji mogu da zauzmu veliki deo memorije. Može vam se desiti da računar prebacuje podatke na disk da bi oslobodio malo memorije za prikaz slika, onda učita deo slika, pa opet prebacuje – i tako nekoliko (hm) minuta. Naročito ako se nalazite na kritičnih četiri megabajta. Takođe su podržani i *Windows* fontovi (naravno, kad program radi pod *Windows*-ima), a u formularu se mogu kombinovati po želji.

Za programiranje pod *Paradox*-om za *Windows* se koristi *ObjectPAL* - jezik koji liči na standardni PAL, ali je obogaćen novim mogućnostima. O ovom jeziku bi imalo šta da se kaže, ali tek kada izađe komercijalna verzija, i uz nju prateća dokumentacija.

Pošto je ovo beta verzija, nismo previše obraćali pažnju na bagove koje je *Borland* spomenuo – uostalom, beta verzija je samo najava izgleda i punih mogućnosti programa. A ova najava nam se dopala.

Kada ovaj paket bude izbrušen i izađe u konačnoj verziji, imaćemo na raspolaganju sjajnu i moćnu alatku za rad sa bazama podataka pod *Windows*-ima.

Kome bi ovaj paket mogao da bude namenjen?

Ako vam je potrebna baza podataka u kojoj se nalaze i slike, a imate 386 i četiri megabajta RAM-a – ovo je dobar izbor. Ukoliko imate potrebnu memoriju, ali ne i 386, bolje nastavite rad pod DOS-om. A ako nemate četiri megabajta, nema vam pomoći – definitivno ne spadate u klasu kojoj je ovaj program namenjen.


NEVEROVATNO
ALI ... ZAISTA
MOGUĆE
intel
25Mhz 386SXL
1,299
DEM
INTERSOFT i386SX SYSTEM
SISTEM SADRŽI 40MB hard disk, 5,25" floppy disk,
14" mono monitor, 1MB RAM
INTERSOFT
29 novembra 43/I,
tel. 323-539, 325-393, fax. 324-120
intel inside


ČAROLIJA STVARANJA
EXPERTS IN COMPUTER GRAPHICS
OSA
RAČUNARSKI INŽENJERING

NOVI IZGLED

AutoCAD je, tokom svojih 11 godina i 11 generacija, izrastao iz malog PC 2-D programa u dominantni svetski standard na polju 2-D i 3-D CAD na svim klasama mikroračunara. Na tržištu se upravo pojavio _AutoCAD Release 12,_ kao i uvek najpre za 386 DOS PC i Sun SparcStation.

Nebojša Novaković

Ljubaznošću ekskluzivnog Autodeskovog distributera, firme „OSA Computer Engineering", imali smo priliku da se detaljnije upoznamo sa beta verzijom *AutoCAD 12* i tako steknemo uvid u najveći deo mnogobrojnih (174 za sada) novosti i unapređenja koja nam je „dvanaestica" donela. Ovaj tekst je ujedno i uvod u veliki test „pravog" programa *AutoCAD 12*, koji možete očekivati u novogodišnjem broju.

KORISNIČKI INTERFEJS

Pored 15 zaista punih instalacionih disketa (beta-verzija zauzima nešto više od 22 MB na disku, a finalna će sigurno i više) i većeg broja priručnika (dodati su *Customizing AutoCAD, AutoCAD SQL, AutoCAD Render* i *Supplemental Programs*, uključujući tu i *PostScript* rad), prva i najprimetnija novost u *AutoCAD 12* je znatno izmenjen i unapređen korisnički interfejs. Po uzoru na *Windows*, *AutoCAD* je prešao na dijalog-prozore, kao i potpunu implementaciju svih naredbi na padajućim menijima, tako da u beta-verziji mnoge od njih već imaju interaktivni rad. Po želji korisnika, stari desni meniji nisu izbačeni, mada novim korisnicima oni ništa ne znače.

Pre novog korisničkog interfejsa, prva novost koja se uopšte primećuje je da nema **Main** menija. *AutoCAD* sam traži drajvere po zadatim direktorijumima i prilikom prvog startovanja nudi izbor svih koje nađe. Primaći ćete i nova drajvera ADI 4.2. Svi su 32-bitni i rade u zaštićenom načinu Intel-ovih 386 i 486 (i P5, takođe). Novi ADI standard omogućuje znatno brži rad periferija, kao i preuzimanje 3-D rada na druge procesore i periferija. To znači da sada bez problema mogu biti korišćeni 3-D miševi i digitalizatori za prostorni unos podataka, kao i posebni brzi procesori tipa i860, za 3-D a ne samo 2-D operacije neposredno u *AutoCAD*-u. Na žalost, naša beta verzija imala je problema sa ADI 4.1 P386 drajverima. S druge strane, ugradnja novog ADI 4.2 386 VESA drajvera (*3D Studio 2* i *Animator Pro* već imaju VESA podršku) znatno olakšava rad korisnicima brojnih VESA-kompatibilnih SuperVGA kartica i to je potez koji u svakom slučaju pozdravljamo. Finalni *AutoCAD 12*, kako smo saznali, ima izuzetnu ugrađenu podršku za veliki broj periferija. Zanimljiva novost je i to da više nema podele na plotere i printer-plotere – sve je pod kategorijom plotera, bio to *Epson LQ 2550, HP LaserJet III* ili pak *HP Draftmaster RX pen ploter*.

Kada obavimo konfiguraciju, automatski ulazimo u **Drawing Editor.** Prva novost koju pritom primećujemo je veliki Autodesk-ov šestarski simbol i podaci o verziji, serijskom broju i korisniku *AutoCAD 12* u levom delu crtaćeg ekrana. Sa ostalim stvarima smo već upoznati: padajući meniji gore, ekranski meniji desno, redovi za naredbe i poruke na dnu ekrana. Međutim, slična spoljašnost ubrzo odaje korenito izmenjenu unutrašnjost.

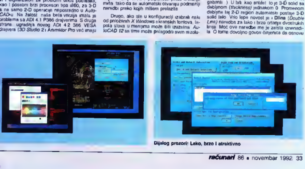
Prvo, padajući meniji su sada organizovani kao i u *Windows* programima: **File, Assist, Draw, Construct, Edit, Settings, View, Render** i **Model**. Fajl meni je najsličniji *Windows*-tajl menijima: opcije **New, Open, Save, Save As, Import, Export, Config** (bivši **Configure AutoCAD** iz **Main** menija), **Utilities, Applications** itd. dovoljno govore. Bivši **Modify** meni sada je razdvojen na **Construct (Array, Copy, Mirror**...) i **Edit (Erase, Move**...) menije, a naredbe **Hide** i **Shade** su iz **View** prešle u **Render** meni, koji je inače kompletan **AutoShade 2** (bez **RenderMan,** koji će se dokupljivati kao dodatna opcija za integraciju). Hijerarhijski padajući meniji sada u potpunosti prate kretanja miša, tako da se automatski otvaraju podmeniji naredbi preko kojih mišem prelazite.

Drugo, ako ste u konfiguraciji izabrali neki od priloženih *X Windows* ekranskih fontova, lepota slova u menijima može biti izuzetna. *AutoCAD 12* se time može prilagoditi svim rezolucijama i veličinama ekrana, tako da meniji uvek budu lako čitljivi.

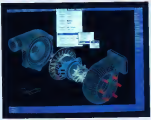
Treće i najznačajnije, najveći deo naredbi se podešava i izvršava u potpunosti preko ljubaznih interaktivnih prozora koji značajno olakšavaju i ubrzavaju rad. Svako ko preko CHPROP u *AutoCAD 11* mora napamet da menja boju ili tip linije objekta, zna na šta mislim. Kod CHPROP u *AutoCAD 12* boju ili tip linije birate neposredno sa spektra u prozoru! Slična unapređenja važe i za većinu ostalih naredbi. Videćemo do kog nivoa je sve to uznapredovalo u finalnom *AutoCAD 12*. Naravno, oni koji to baš žele i dalje mogu ukucavati naredbe i njihove opcije klasičnim putem. Međutim, rad sa dijalog-prozorima je toliko brz, lak i, zašto ne, atraktivan da zaista ne treba gubiti vreme sa komandnom linijom – mani kao i svima koji baš ne vole komandne linije, ovo unapređenje došlo je kao poručeno. A kako li tek izgleda korisnički interfejs *AutoCAD*-a *12* na *Windows NT, Sun SparcStation* ili *IRIS Indigo*.

NOVE NAREDBE

AutoCAD 12 donosi velika unapređenja u odnosu na prethodnu verziju i na polju naredbi. Pošto u testiranoj beta-verziji još nisu implementirana sva najavljena poboljšanja, ovom prilikom ćemo se osvrnuti samo na najznačajnije novosti, kojih je inače dosta i na 2-D i na 3-D polju. Počnimo od ravanskog crtanja. Iz AME 2 za crtače stiže lepa novost: **Region Modeler** - 2-D ekvivalent 3-D solid modelera. Tako sada možete vršiti logičke operacije i na 2-D elementima (na primer, presek kruga i mnogougla, unija elipse i kvadrata itd.) gde rezultat dobija sve potrebne parametre (površina, gabariti...). U biti, kao antitet, to je 3-D solid sa debljinom (*thickness*) jednakom 0. Promenom debljine taj 2-D region automatski postaje 3-D solid telo. Vrlo lepa novost je i **Dline** (*Double Line*) naredba za lako i brzo crtanje dvostrukih linija. Moć ove naredbe me je zaista iznenadila. O tome dovoljno govori činjenica da osnovu



Dijalog prozori: Lako, brzo i atraktivno



Nova struktura menija. Po uzoru na *Windows*, *AutoCAD* je prešao na dijalog-prozore, kao i potpunu implementaciju svih naredbi na padajućim menijima



Vizualizacija *AutoCAD 12* solid modela uz pomoć programa *RenderMan*

zidova kuće sada možete iscrtati maltene samo jednom **Dlina** naredbom. Preko **Dlset** dijalog-prozora podešavate sve parametre **Dlina**.

Još jedno bitno unapređenje je novi način šrafiranja, *Boundary Hatching* naredbom **Bhatch**. Kao što mu ime kaže, **Bhatch** se zasniva na bacanju zraka (*Ray Casting*) iz unutrašnjosti površina na ivice (*boundaries*). Prednosti su znatno veća brzina šrafiranja, mnogo lakše podešavanje i skaliranje šrafura, što uz ljubazan interaktivni prozor i mogućnost neposrednog unosa i korišćenja *PostScript* šrafura donosi sasvim novi kvalitet rada.

Jedno od glavnih unapređenja u *AutoCAD 12* je upravo vezano za *PostScript*. Naime, *AutoCAD* sada ne samo što prihvata *PostScript* crteže i daje *PostScript* izlaz, nego neposredno prihvata i kompajlira sva *Adobe Type 1 PostScript* fontove! To, na primer, znači da možete uzeti svih 250 *CorelDRAW 3.0 PostScript* fontova sa njegovog CD-ROM, koristiti ih u *AutoCAD*, i pored sasvim novog izgleda vaših crteža tim istim fontovima dati i treću dimenziju! *3D Studio 2* takođe neposredno prima *PostScript* fontove, dajući još veće mogućnosti 3-D manipulacije, vizualizacije i materijalizacije tih, često ekstravagantnih, neobičnih ili jednostavno lepih fontova.

Kad smo već kod CD-ROM, treba pomenuti da se i uz prodajnu verziju *AutoCAD 12* takođe dobija CD-ROM disk prepun svega i svačega (neko bi rekao da bi za cenu po kojoj prodaju *AutoCAD* usput mogli da daju i CD-ROM drajv). Ako taj CD stavite u klasičan audio CD plejer, čućete šta na njemu ima. A o tome šta ima više u testu „pravog" *AutoCAD 12*.

Na 3-D polju glavni naglasak stavljen je na poboljšanu manipulaciju 3-D elementima preko naredbi **Align3D, Mirror3D** i **Rotate3D.** Kao što vidite, sada 3D element možete poravnati sa bilo kojom ravni ili objektom, reflektovati ga kao u ogledalu preko bilo koje ravni i rotirati ga oko bilo koje ose u prostoru. Naredbom **Project3D** nacrtna geometrija je pred vama – projekcija 3D elemenata na bilo koju ravan u prostoru.

Novi **Render** meni, pored kompletnog **AutoShade 2** (**RenderMan** je posebna opcija koja se dokupljuje), sadrži i stara naredba za brzo *AutoCAD* senčenje (**Shade**) kao i **Hide** naredbu za skrivanje nevidljivih linija. Ovo je zaista novost. **Hide** na *AutoCAD 12* je od **Hide** na *AutoCAD* 11 brži i do 100 puta! Ubrzanje je stvarno – ono što 11 uradi za minut, 12 uradi za sekund.

AME je takođe doživeo ceo niz poboljšanja, od 64-bitne preciznosti koja eliminiše sve potencijalne greške u operacijama do lakog i brzog vađenja preseka i projekcija solid modela.

Ova poboljšanja u 3-D radu nesumnjivo ukazuju na dalju orijentaciju ne samo *AutoCAD*-a već i celokupnog CAD-a u računarstvu na puno 3-D modeliranje u svim zadacima, kao i postupno povlačenje 2-D crtanja na ona retka područja gde 3-D rad baš nema nikakve svrhe. To ujedno znači da ljudi u ovoj oblasti treba konačno da prihvate novu logiku 3-D modeliranja i 2-D crtaža samo kao naknadno izvađenih preseka 3-D modela smeštenih u *Paper Space* - koordinatni prostor hartije na izlaznom uređaju.

Ono što još očigledno nedostaje da *AutoCAD 12* bude san snova je nedostatak NURBS (*Non-Uniform Rational B-Spline*) krivih koje su neophodne za, recimo, dizajn ozbiljnijih automobilskih karoserija, ali modularna struktura *AutoCAD* omogućava dodavanje i ovih, kao i drugih operacija, bilo kao eksternih naredbi koje se po želji automatski učitavaju u *AutoCAD* po startovanju programa, ili kao dodatnih aplikacija koje rade u *AutoCAD* okruženju.

U beta verziji mogli smo videti još dve lepe novosti koje *AutoCAD 12* donosi: kalkulator i tekst-procesor. **Cal** eksterna naredba pruža nam lep skalarni i vektorski inženjerski kalkulator sa nizom operacija. Ove operacije se mogu upotrebljavati u jednačinama prilikom zadavanja promenljivih u naredbama. Na primer, jedna tačka linije može imati koordinate zadate kao koren logaritma nekog broja, inverzni sinus skalarnog proizvoda dva vektora i recipročnu vrednost nekog broja za X, Y i Z ose respektivno, a druga tačka može, na primer, biti vektorski proizvod neka dva vektora. **Ptekt** naredba daje mali i prost tekst-procesor za uređivanje *AutoCAD* teksta u crtežu. Broj i moć operacija otprilike odgovara *Quick Edit*-u, s tim što je ovde reč o editoru vektorskih slova koja su ipak nešto posebno. Inače, ukoliko imate grafiku rezolucije 800x600 ili više, *AutoCAD 12* omogućava vam da tekst-prozor, za koji je ranije bio potreban skok u ružni VGA tekst-mod, sada gledate zaista kao tekst-prozor na crtaćem ekranu – lepa novost.

UKUPNE PERFORMANSE

AutoCAD 12 donosi ubrzanje i istovremeno olakšanje rada u većini naredbi. Nesumnjivo je da će produktivnost rada sa „dvanaesticom" biti znatno veća, pogotovo kod PC računara sa ultrabrzim inteligentnim grafičkim karticama. Ugrađena 32-bitna ekranska lista, sa gotovo neograničenim faktorima zuma i pena, mučni REGEN šalje u zaborav. Ukoliko ta ekranska lista zasebno radi na procesorima kao što su TI 34020 ili još brži i860, brzine iscrtavanja i drugih ekranskih operacija su trenutne i kod najkompleksnijih crteža. A kod 860 i sličnih 3D procesora gledaćemo 3D ekranske liste, a kojima će ionako brze *AutoCAD 12* 3D operacije postati još višestruko brže, naročito kod senčenja.

Naravno, stepen ubrzanja zavisi i od hardvera na kome se cela stvar vrti, a minimum je 486DX-33, dok je koliko-toliko podnošljiva konfiguracija 486DX-50 ili 486DX2-66 sa velikim sekundarnim *write-back* kešom, barem 16 MB brzog 60 ns RAM, 240 MB diskom i *local-bus* ili koprocesorskom grafikom, sa obaveznom ADI 4.2 podrškom. Nezaobilazna činjenica je takođe da PC, pa makar i ovakav bio, nije ni do kolena ni najjeftinijoj ozbiljnoj RISC radnoj stanici tipa *Sun SparcStation IPX* ili *SGI IRIS Indigo*. Jednostavno, zna se kakvi računari odgovaraju grafičkim aplikacijama. Na kraju, RISC više i nije tako skup – cene su često i niže od cena malo ozbiljnijih 486 mašina. Sve u svemu, optimalna *price/performance* platforma za CAD je *AutoCAD 12* koji gospodski radi na nekoj lepoj radnoj stanici, pa izvolite, birajte: Sun, Silicon Graphics, HP, DEC, IBM.

Korisna adresa

OSA RAČUNARSKI INŽENJERING
11000 Beograd, Šumadijski trg 6a
Tel. 011/557-992, Fax 011/559-845

NAJZAD PRAVA STVAR

Da nismo baš u potpunoj izolaciji svedoči i ova kopija *Quattro Pro for Windows* koju smo nedavno dobili. Radi se o beta verziji ovog programa koji je stao na četiri diskete od 3.5", 1.44 Mb i uz koji, na žalost, nema nikakve dokumentacije. To nam, međutim, nije smetalo da ga pažljivo pregledamo, jer se radi o programu koga korisnici ove vrste softvera iščekuju s velikim nestrpljenjem.

Vladimir Stamenović

Pre nego krenem na instalaciju programa treba reći da je *Quattro Pro for Windows* vrlo zahtevan u pogledu hardvera. Nadam se da tako neće biti i u finalnoj verziji, ali sada je trebalo obezbediti računar sa više od 4 Mb RAM-a, dok je na disku neophodno oko 15 Mb slobodnog prostora! U suprotnom, od instalacije nema ništa. Interesantno je da nakon instalacije *Quattro Pro for Windows* zauzima, ukupno, oko 8 Mb. Onaj zahtevani višak od oko 7 Mb stavljam na dušu instalacionom programu. On je urađen veoma atraktivno, pokreće se iz *Windows*-a, ali je veoma, veoma spor. Da bi raspakovao i prebacio datoteke sa četiri disketa na disk, potrebno mu je više od 35 minuta.

KONCEPT BELEŽNICE

Nakon završene instalacije možete pokrenuti program i sačekaće vas dosta prijatnih iznenađenja. Naime, u ovakvom i sličnim tipovima programa nisam video lepši i atraktivniji interfejs. Središnji i najveći deo ekrana, razumljivo, zauzima tabela. Ona se sada zove beležnica *(Notebook)*, a svaka ima svoj naziv, kao NOTEBK1.WB1, NOTEBK2.WB1..., što ne smeta da tabelu snimite pod nekim svojim imenom. Možda je najzanimljivija novost u programu da je prihvaćen Lotusov koncept trodimenzionalne tabele (iz verzije 3.xx). Naime, svaki *Notebook* ima do 256 tabela i jednu grafičku stranu. Sve to zajedno se smešta u jednu datoteku! Vrlo je lepo rešenje, u interfejsu za promenu aktivne tabele. One su poređane jedna iza druge, a svaka ima izrez kao u rokovniku, obeležen nazivom tabele (od A do IV). Vidljivo je samo sedam prvih naziva tabela, dok za ostale postoji, u donjem levom uglu, mali skrol-bar pomoću koga se može doći i do ostalih tabela. Brzi prelaz iz aktivne tabele u poslednju i obrnuto se ostvaruje klikom na **SpeedTab** dugme koje se nalazi u sredini donjeg reda na ekranu, desno od oznake tabela, a obeleženo je strelicom levo ili desno, zavisno od toga na koju stranu se krećete. Za prelaz iz jedne u drugu tabelu, dovoljno je da mišem kliknete na odgovarajuću oznaku. U levoj polovini donjeg reda ispod tabele i sa leve strane tabele nalaze se horizontalni i vertikalni skrol-barovi, koji kao i kod svih *Windows* programa, služe za kretanje po tabeli.

Iznad tabele, u prvom redu, nalaze se komande za rad sa programom. Njihov raspored, aktiviranje, kao i skraćenice za rad su uobičajeni u ovakvoj vrsti programa. Prve po redu je **File** komanda pomoću koje se barata datotekama, štampaju dokumenti i izlazi iz programa. Sledeća je **Edit** komanda koja omogućuje rad sa *Clipboard*-om (**Cut, Copy, Paste**), a sadrži i komanda **Goto, Undo** i **Search**. Za rearanžiranje tabele služi komanda **Block** (**Move, Copy, Fill, Reformat, Insert** i **Delete** kolona, redova i strana). Rad sa bazom podataka se odvija kroz komandu **Data**, a veoma zanimljive mogućnosti nudi komanda **Tools**. U njen domen spadaju rad sa makroima, kombinovanje i linkovanje datoteka, rad sa matricama, „šta-ako" analize, *Solver* i mogućnost promene izgleda meni bara *(UI building)*. Grafičku prezentaciju podataka ćete raditi kroz komandu **Graph**. Tu možete kreirati, doterivati i posmatrati grafikone i tzv. *Slide show*, kao i ubacivati gotove sličice (*clip art*). Komandom **Property** možete vršiti razna podešavanja za svaku vrstu objekta, dok za rearanžiranje prozora služi komanda **Window**. Na kraju menija je komanda **Help**.

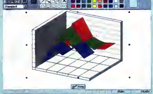
BRZO PO TABELI

Ispod menija nalazi se bar sa nizom različitih ikona koje bi trebalo da ubrzaju najčešće obavljane poslove u tabeli (pun naziv je *Notebook Window SpeedBar*). Ima ih dvadesetak. Prve tri služe umesto komandi **Cut, Copy** i **Paste**. Podaci iz ćelija ili blokova se pomoću ikona vrlo brzo mogu isecati, kopirati i premeštati bilo gde u tabeli, u sve tri dimenzije. Tu je i ikona za ubacivanje simbola za dugmad kojima će se izvršavati neke makro procedure. Brzo kreiranje grafikona, poravnavanje teksta na sva tri načina, promena formata teksta (*bold* i *italics*) i brojčanih podataka je samo deo poslova koji su znatno ubrzani preko ikona. Postoji i ikona koja omogućuje brzo sabiranje, promenu visine reda ili, vrlo interesantna, *SpeedFormat* ikona, koja otvara dijalog-boks sa pet predefinisanih formata za tabelu. To znači da je već određeno kako će tabela izgledati: sa iscrtanim linijama ispod zaglavlja, osenčenim važnijim blokovima, različitim tipovima fontova... Impresivno! Treba reći da se bar sa ikonama menja u zavisnosti od posla koji se radi. Recimo, ako editujete grafikon, tu su potpuno nove ikone koje omogućuju veoma komforan i lagan rad na doterivanju izgleda grafikona.

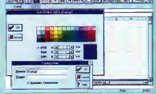
U trećoj liniji ekrana je smešten *edit bar* (*Edit Mode SpeedBar*). U njegovom levom delu se vidi adresa aktivne ćelije ili bloka, a ako pritisnete taster **F2**, onda se u nastavku vidi i sadržaj koji se može korigovati. Pri unosu podataka od pomoći su ikone koje ubrzavaju posao, na primer oko ispisivanje makro komandi (zajedno sa vitičastim zagradama).

Nadam se da ste već mogli da zaključite da su *Borland*-ovi programeri učinili veliki napor da korisnicima omoguće komforan i brz rad. Čini mi se da su u tome uspeli.

S obzirom da je ova verzija *Quattro Pro for Windows* instalirana na brzom 386 računaru, program je radio veoma brzo. Ni traga od nepoželjnih i sporih osvežavanja ekrana. Ikone su brzo odgovarale na aktiviranje, pa je rad bio pravo zadovoljstvo. Šteta je što uz program nemamo nikakvu dokumentaciju, a da stvar bude još gora, *Help* je nedorađen, pa se često javlja poruka da se određena strana *Help*-a odnosi na QP 4.0 za DOS ili jednostavno nema



Lako i zabavno: Kreiranje i doterivanje grafikona



Dijalog editor: Kreiranje i povezivanje dijalog-boksova

nikakvog teksta osim naslova. No i bez toga je jasno da se radi o kvalitetnom proizvodu.

PET RADNIH PROSTORA

Quattro Pro for Windows ima pet radnih prostora. Prvi je *Notebook Window*, prostor gde unosimo i prikazujemo podatke, a mi ga nazivamo tabela. Sadrži sve standardne mogućnosti *spreadsheet* programa, ali i mnoga proširenja i dodatke. U njemu se obavlja najveći deo posla. Kreiranje, doterivanje i ispravka grafikona, baziranih na podacima iz tabele, obavljaju se u *Graph Window*. Ako pravite sopstvenu aplikaciju koja će raditi u *Quattro Pro for Windows* koristićete *Dialog Window*. *Graphs Page* je poslednja strana u svakoj tabeli (*Notebook*). Tu vršite povezivanja grafikona i slika u *slide show* ili povezujete sopstvene dijalog-boksova koji će činiti neku aplikaciju. Peti radni prostor je *Print Preview Screen* gde se može pregledati rad pre štampanja.

Da se vratimo tabeli. Dakle, kao što sam rekao, imamo trodimenzionalnu tabelu. Svako ko je ikada radio sa *Lotusovim 1-2-3* verzija 3.xx zna kakva je to prednost u odnosu na „običnu" tabelu. Primera radi, ako imate nekoliko tabela koje služe za obavljanje sličnih poslova ili su čak međusobno linkovane, onda je daleko čistija situacija ako svaku od tih tabela ubacite u poseban list u jednom *Notebook*-u. Tu je povezivanje podataka daleko lakše, a rad znatno brži. Na kraju krajeva, umesto nekoliko datoteka imaćete samo jednu. Poslednja strana u *Notebook*-u je grafička strana. Ona sadrži ikone koje predstavljaju svaki od napravljenih grafikona, *slide show*-a ili dijalog-boksova koje ste sami kreirali. Karakteristično za *Quattro Pro for Windows* je da gotovo svi elementi koji se vide na ekranu predstavljaju objekte. Svaki od njih se može posebno podešavati i doterivati. Na primer, u objektu koji se zove *Aktivni blok* možete vršiti najrazličitija podešavanja na aktivnoj ćeliji ili bloku, ili to isto ali na *Graph* objektu.

Kao u verziji QP 4.0 za DOS, i ovde mi se najviše dopao rad u grafičkom prozoru. Pravo je zadovoljstvo kreiranje i doterivanje grafikona, mešanje sa gotovim sličicama (probao sam samo CGM format) ili njihovo prepravljanje. Maštovito, pažljivo i vrlo kvalitetno je urađen ovaj deo programa. Kada editujete grafikon ili sliku, u posebnom prozoru u *SpeedBar*-u dobijete nov set ikona, a meni ima nekoliko novih komandi. Posao je uglavnom jednostavan. Klikom odaberete objekt, a zatim preko ikona ili kroz meni vršite odgovarajuće doterivanje. Dijalog-boksovi su impresivni, na višem nivou od onih u *MS Excel*-u. U njima se vidi reakcija na svaku promenu u podešavanjima nekog objekta. Posebno me zainteresovala mogućnost da prepravljam gotove sličice. Načini na koji se to može uraditi gotovo da nemaju broja: promena boja i nijansi svih elemenata na slici, promena šrafure i punoće boja, debljina linija, dodavanje novih elemenata u sliku itd... Vrlo lako i zabavno.

POBOLJAN U VIŠE PRAVACA

Quattro Pro for Windows je poboljšan u više pravaca u odnosu na proizvode za DOS. Recimo, ima dosta novih @ funkcija tako da ih je sada ukupno oko 118, što ne znači da će to, u finalnoj verziji biti konačan broj. Izdeljene su u nekoliko grupa: matematička, statistička, za baze podataka, logička, finansijska, za datume i vreme i string funkcije. Nema sumnje da će ovoliki broj dobro osmišljenih funkcija doprineti da korisnici prave dobre i kompleksne tabele za najrazličitije poslove.

Što se tiče kompatibilnosti sa *1-2-3*, izgleda da je sve u redu i da *Quattro Pro for Windows* prihvata sve makroe i @ funkcije iz *1-2-3* verzija 2.xx i 3.xx. To bi trebalo da bude veliko olakšanje onim korisnicima *1-2-3* koji se eventualno reše da pređu na rad sa *Quattro Pro for Windows*. Prilikom ubacivanja makroa iz *Lotusovih* tabela ne treba zaboraviti da se mora aktivirati takozvani *1-2-3 menu tree*, a za upotrebu uobičajenih komandi u *1-2-3* i *QP* za DOS treba aktivirati *Key Rider*. Makroi koji se u *1-2-3* pokreću uz pomoć **Alt** ovde će se startovati pomoću **Ctrl** tastera.

U ovoj beta verziji *Quattro Pro for Windows* ima zadivljujućih 149 makro komandi. Izgleda da su Borlandovi programeri mislili na sve, jer prosto nema onoga posla koji se pomoću njih ne može uraditi. Ovo ocenjujem kao veliki napredak u *spreadsheet* programima. Kod pisanja makro procedura od velike pomoći je ugrađeni *Macro Debugger*, pomoću koga se makroi mogu kontrolisati na nekoliko načina: korak po korak, postavljanjem tačaka prekida kada makro staje ako je zadovoljio određeni rezultat ili uslov, pokretanjem makroa punom brzinom do tačke prekida, a onda korak po korak. Rad sa dibagerom se odvija u posebnom *Debug window*.

Veliki kvalitet *Quattro Pro for Windows* je da ga korisnik može sebi maksimalno prilagoditi. Recimo, ne morate koristiti standardne *Windows* meni komande, već to možete raditi iz nekog od alternativnih (*1-2-3* ili *QP*) menija koje ćete pozivati sa „/". Meni bar možete zameniti sopstvenim menijem, koji će sadržati vama potrebne komande. Za to treba iskoristiti mogućnosti koje se nude u alatima za izradu sopstvenih aplikacija. U ovom programu aplikacija je mešavina i kombinacija makroa, dijalog-boksova i menija. Za izradu aplikacija postoje alati takozvani *Application Tools*. Pomoću njih možete doterati *Quattro Pro* tako da ispunjava specijalne zahteve korisnika: možete isključiti nepotrebne komande, čime interfejs postaje jednostavniji (setite se „Batera" za *1-2-3*), a postoji i mogućnost kreiranja interfejsa za nove, vaše, komande. Nisam stigao da sve te mogućnosti isprobam, ali gledajući u jednu malu aplikaciju koja se nalazi na disketi, čini mi se da je Borland napravio pravu stvar.

I pored toga što se radi o beta verziji, dakle o nezavršenom programu, ovo je, po mojoj oceni, najbolji *spreadsheet* koji radi pod *Windows*-ima. Ostavio je na mene bolji utisak od *MS Excel*-a, mada nisam stigao da ga do detalja isprtam. *Quattro Pro for Windows* ima fantastičan interfejs, manipulacija objektima je izuzetna, kretanje po trodimenzionalnoj tabeli je veoma lako, a ovoliki broj i ovakve @ funkcija i makro komanda, te sistem građenja aplikacija garantuju da će korisnici imati velike mogućnosti u pravljenju sopstvenih aplikacija. Neki manji problemi u radu programa, a tu pre svega mislim na *Help* fontova i *Preview* mod, će sigurno biti prevaziđeni u finalnoj verziji. Ako konačna verzija bude nešto manje zahtevna u pogledu hardvera (min. 4Mb RAM-a) *Quattro Pro for Windows* će biti jedan od najboljih, ako ne i najbolji *spreadsheet* na tržištu.

SAJMOVI

Beograd: COMPUTERS 92

Beogradski market računara

COMPUTERS 92 – market za profesionalnu računarsku opremu i prateće sadržaje – rezultat je želje da se, u nedostatku adekvatnih mogućnosti za predstavljanja računarskih proizvoda, organizuje kvalitetna manifestacija za informatiku, poput nekadašnjeg INTERBIRO-a u Zagrebu

Inicijator ovog marketa, preduzeće MARCON AG, je pozvalo sve potencijalne izlagače na sastanak u hotelu Beograd Inter-Continental, znajući da su njihove sugestije dragocene i da predstavljaju osnov za dobru koncepciju i organizaciju marketa. Sastanku su prisustvovali predstavnici 20 kompjuterskih firmi i svi su se saglasili da izlaganje svedeno na „Beogradski sajam" ne zadovoljava u potpunosti njihove potrebe.

Bitan deo razgovora posvećen je pratećim manifestacijama. Planirane su prezentacije i promocije (programa, proizvoda, sistema...) za koje se javljaju izlagači prema svojim interesima, a onda i seminari, tribine i okrugli stolovi vezani za specifične probleme i potrebe izlagača. Na ove razgovore bili bi pozvani i predstavnici saveznih i republičkih organa – naravno, oni koje izlagač zaželi da vide.

Dakle, za razliku od klasičnih sajamskih postavki, COMPUTERS '92 će biti začinjen i raznovrsnim pratećim sadržajima koji će obuhvatiti sve trenutno interesantnije oblasti: mreže, profesionalnu grafiku (težište na primeni), poslovne programe, poslovne sisteme i komunikaciju, multimediju, periferne uređaje (hardver i softver), primenu vizuelizacije, simulacije i animacija.

Naravno, neće izostati ni klasični izlagačko-prodajni aspekt. Na jednom mestu biće predstavljeni najnoviji modeli uređaja, tehničke inovacije i najnovije korisničke mogućnosti postojećih sistema – jednom rečju, sve što se može nabaviti na našem tržištu.

Za mesto održavanja odabran je hotel Beograd Inter-Continental, koji su svojim prostranim dvoranama, konferencijskim i projekcionim salama pruža izuzetne mogućnosti za ovakvu manifestaciju, kao i ugodnu atmosferu za sklapanje poslovnih aranžmana.

Dogovoreno je da market bude „zatvorenog" tipa, jer je namenjen isključivo poslovnom svetu – profesionalni informatičari, izlagači i posetioci će na taj način dobiti najbolje uslove za kontakte. Svi zainteresovani se mogu prijaviti organizatoru, uključujući i posetioce koji se registruju i uplaćuju kotizaciju, ako se jave pre početka marketa **4.000 dinara** po osobi za sve vreme trajanja, a u vreme trajanja marketa **3.000 dinara** po osobi dnevno.

Mnogima su već poznate tradicionalne manifestacije u Beogradu – VIDEO SHOW, VIDEO SCREEN, COMPUTER SHOW – koje su organizovali ljudi iz preduzeća MARCON AG. Dokazali su da taj posao dobro rade, pa im zaželimo uspeh i ovaj put.

Sve firme koje žele da učestvuju na marketu kao izlagači ili posetioci mogu se javiti direktno organizatorima da bi dobili opširnije informacije o uslovima prijavljivanja.

Kontakt adresa: Gordana Hajnović – organizator i Dragutin Gaga Marković – direktor, tel. 011/327-142, tel/fax 324-640, Lole Ribara 45, 11000 Beograd.

PRIJAVA ZA COMPUTERS '92
Beograd, 17. – 19. novembar 1992.

Učesnik ______________________

Sedište (mesto, ulica i broj) ______________________

____

Telefon ____________ Fax ____________

Osoba za kontakt ______________________

Molimo da nam obezbedite mogućnost učešća na COMPUTERS '92.
Potrebno nam je ________ (navesti broj) ID kartica na ime

1. ____________ 2. ____________

3. ____________ 4. ____________

U prilogu dostavljamo potvrdu o uplaćenoj kotizaciji.
Uplata na žiro račun 60802 601-6894 SDK Beograd
Primalac: MARCON AG, Lole Ribara 45, 11000 BEOGRAD, svrha doznake COMPUTERS '92

Kopiju uplatnice i prijavu poslati organizatorima poštom ili faxom

NEKA BUDE NOVEL

Nema sumnje da je *Novell* kralj među mrežnim programima. Ima i drugih dobrih programa, ali, kako god gledali na stvar, *Novell* je uspeo da postane referentni standard. Da li će to poći za rukom i programu *Novell Lite* – krajnje pojednostavljenom proizvodu za male mreže sa najskromnijim zahtevima?

Dejan V. Veselinović

Verzija 3.11 je evoluirala iz druge generacije mrežnih programa koji su definitivno učvrstili firmu Novell kao apsolutnog lidera na ovom delu tržišta. Biti lider ne znači imati bezuslovno tehnički najbolji proizvod, ali sasvim sigurno znači imati najveći obim prodaje, što podrazumeva i velike prihode, koji povlače velike budžete za istraživanja.

Ali, nema tih prihoda koji garantuju imunitet na greške, pa makar i relativne (onda to postaju propusti). Kao i većina američkih firmi, firma *Novell* je odmah mislila u relacijama velikog, pa je svoje programe uglavnom predvidela za velike mreže. "Veliko" je sve od dvadesetak mašina u mreži pa naviše, a stiglo se i do preko 250 jedinica u mreži. To je divno za velike organizacije kojima to treba i neće im smetati nimalo skromna cena programa. Međutim, to uopšte nije zgodno za manje i male organizacije, koje nemaju niti će u dogledno vreme imati više od desetak mašina u celoj mreži. Kada treba da povežete samo tri ili četiri mašine, cena celog paketa mrežnog programa bi se mogla maltene izjednačiti sa svim nodovima (čvorištima) u mreži. A gde su problemi oko postavljenja cele mreže, što je donedavno, zbog komplikovanosti, bilo izvor unosne zarade (a ni danas nismo daleko od toga).

Sve ovo su uočile male i na tržištu nove firme, kao što je donedavno nepoznata firma *Artisoft*, koja je svojim mrežnim programom *LANtastic* veoma uspešno probila barijeru cena i složenosti. Uprkos relativno velikom broju podržanih mašina, sami autori jasno kažu da je njihov program napravljen za male mreže, odnosno za mreže sa najviše 20 ili 25 nodova. Osim toga, cena njihovog programa je bila neuporediva sa *Novell*-om – iznosila je šestinu ili sedminu cene *Novell*-a, i to zajedno sa potrebnim hardverom.

Pre otprilike dve godine, *Novell* se superiorno smeškao i tvrdio da je sve to prolazna moda (usput pregovarajući sa firmom *Lotus* o ujedinjenju, što je propalo), a onda je neki od njihovih knjigovođa seo i sračunao koliko oni relativno i apsolutno gube na toj "modi", a firme kao Artisoft zarađuju. Direktora mora da je udarila kap kada je video brojke, jer je posle toga neko pritisnuo taster za paniku. Rezultat je program nazvan *Lite* (nemamo pojednostavljeno napisano umesto "Light" – lako, lagano ili pak svetlo ili svetlost). Poenta je da se radi o pojednostavljenom mrežnom programu koji za veoma prihvatljive pare omogućava umrežavanje mašina i, naravno, konkuriše softverskim mladoturcima.

NEKOLIKO OSNOVNIH NAČELA

Ne upuštajući se u dugačke rasprave o vrstama, podvrstama, cenama, kvalitetu i poreklu raznih mogućnosti mreža, ipak smatramo da je potrebno navesti nekoliko osobina mreža koje podjednako važe za sve.

- **Mreža mora biti pouzdana.** Ne sme svaki čas padati, niti izbacivati razne poruke sa opisima grešaka. Pouzdanost je daleko najvažnije osobina mreže – to je jedina osobina na kojoj ne sme biti kompromisa ili nagađanja.
- **Mreža mora biti tolerantna.** Pod tolerancijom se misli i na toleranciju prema greškama korisnika i na odabranu mašinsku podlogu. Jednostavne greške korisnika ne smeju tek tako obarati mrežu, jer će se greška stalno događati. Mreža ne sme biti suviše probirljiva u kontekstu hardvera, uvek će biti delova sistema koji će biti problematični, ali njihov broj treba da bude veoma mali i samim tim redak u praksi. Velika probirljivost najčešće znači i malu pouzdanost.
- **Mreža mora biti fleksibilna.** Drugim rečima, mora omogućavati naknadnu dogradnju bez većih problema, kako u smislu novih mašinskih delova sistema (na primer, dodavanje novog diska), tako i u smislu dodavanja novih čvorišta (nodova). Nije dobro ako zbog najmanje izmene morate da instalirate sve od početka.
- **Mreža mora imati dobru podršku.** To ujedno znači dobru dokumentaciju i razrađene mehanizme nadogradnje i, što je najvažnije, poznat, pouzdan i stručno osposobljen izvor odnosno prodavca. Mreže su ozbiljne stvari i nisu za igranje, a autori su veoma, veoma daleko, dakle sve što vam ostaje od aktivne podrške jeste prodavac. Ako on dobro radi svoj posao, sve novine će stići od vas i bez upita, a uvek će biti nekoga da vam se nađe oko eventualnih problema. Istina je da ima veoma dobrih ljudi koji nisu zvanični prodavci ove ili one firme, ali, ako imate izbor, uvek se opredelite za zvaničnog zastupnika ili prodavca, posebno ako može da vam pokaže dokaze o pohađanju kurseva za mrežne programe.
- **Mreža mora biti cenovno opravdana.** To znači da cena umrežavanja mora da bude manja od cene prostog zbira ekvivalentnih računara ili, u najgorem slučaju, ista. Ima situacija kada je usled prirode posla umrežavanje neminovnost bez obzira na cenu, ali je u većini slučajeva moguće projektovati mrežu tako da bude i relativno i apsolutno jeftinije od prostog zbira računara sa sličnim efektom. Osim toga, ako se postigne niža cena, to znači da sve dobiti od umrežavanja predstavlja vašu ekstra dobit – a baš tako i treba da bude.

Odavde je veoma lako zaključiti kako treba pristupiti uvođenju mreže u organizaciju:

- Pažljivo isplanirajte vaše sadašnje i poznate buduće potrebe;
- Pažljivo odaberite izvor delova i opreme, sa akcentom na podršci;
- Nemojte štedeti na čisto mrežnom hardveru, jer on vrši posao ili pada;
- Dobro proračunajte nekoliko mogućih varijanti umrežavanja.

ŠTA JE SVE POTREBNO

Program je zaista "lagan", i po obimu i po ceni, ceo program staje u jednu kutiju veličine nešto malo deblje knjige, a umesto nekih DEM 3 500-4 500 kao od sada, treba da odvojite nekih DEM 300 po računaru. Za mrežu od, recimo, 5 računara, sada plaćate jedva trećinu cene "velikog" programa.

Uz *Novell Lite* vam treba onoliko mrežnih (Ethernet) kartica koliko i mašina u mreži. Tu smo naišli na prvi problem – ne zbog vrste kartica (znalo se odmah: 16-bitne sve od reda) već zbog njihovog proizvođača, a samim tim i cene. Mogli smo se opredeliti za totalno anonimne kartice iz nepoznatog izvora (sem geografski – Tajvana), po pristupačnoj ceni, ali nepoznatog kvaliteta, alternativa je bila uzeti poznate proizvode, iz cenjenog izvora, po srazmerno većoj ceni. Mi smo se opredelili za drugu varijantu i uzeli dve Western Digital "Elite16" kartice, jer važe za veoma pouzdane, čiste proizvode, koji rade sa zadovoljavajućim brzinama (iako nam ovo nije bio poseban problem, s obzirom na malu mrežu). Naravno, i na žalost, radi se i o dosta skupim proizvodima, sa razlikom u ceni u odnosu na manje poznate i od 2:1.

Kartica ima obe vrste veza, kako za paralelni tako i za koaksijalni kabl, mi smo koristili koaksijalnu (BNC) vezu, koja je praktičnija a i kod nas se najčešće koristi (čak se ni ne sećamo da smo ikada videli onu drugu vezu u upotrebi). Kartice su na uobičajeno visokom nivou ove kuće, sa površinskim lemom, fantastičnom štampom i, uopšte, odličnom izvedbom. Paket u kome se isporučuju je zaista sveobuhvatan, pored uobičajene razvodne "T" veze, tu je i 50-omski terminator, knjižica sa uputstvom i disketa sa veznim programima. Na kartici postoji podnožje namenjeno ROM-u za podizanje mašine (boot ROM).

Priručnik je retko dobro napravljen. Izuzetno je informativan, veoma pregledan i, sa detaljnim i jasnim uputstvima, pogodan čak i za apsolutne početnike. Za one iskusnije, opis svrhe kratkospojnika je bojom nanesen i na samu karticu, pa njima priručnik nije ni potreban.

I tako dođosmo do samog *Novell*-a. I on može poslužiti za primer po kvalitetu dokumentacije. Jedino može da zasmeta to što su ga autori namenili potpunim neznalicama i pojednostavili do nivoa deteta. Zato se dešava da dugo tražite ono što vam treba, mada se, istini za volju, takva potreba retko ukazuje.

INSTALACIJA

Preskočićemo opisivanje ugradnje kartica. Napominjemo da smo malo varali prilikom instalacije, utoliko što smo sa jedne strane, umesto uzemljenog terminatora od 50 Oma, koristili nejobičniji terminator koji se isporučuje uz karticu. Nije preporučljiva praksa, ali, eto, hoće da radi.

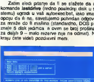
Lite se sastoji od dve diskete formata 5,25 inča i gustine zapisa od 360 KB (ili jedne di-

skete od 3,5 inča i 720 KB). Instalacija je jednostavna – stavite disketu označenu sa naslovom "Program" u disk jedinicu A:, pritisnete Enter i sve će krenuti samo od sebe. Ispred vas će se pojavi naredni meni



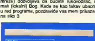
Nakon kraćeg vremena, na ekranu će se pojaviti podmeniji sa očekivanom ponudom izbora, treba da navedete koju mrežnu karticu koristite (spisak je veoma dugačak – eiai vere – a ima i opciju **Other**, tj. "druga" za slučaj da vaš model nije naveden), a zatim potvrdite ili izmenite standardne parametre podešavanja (određivanje interapta, adrese bazne memorije i I/O adrese, tri elementa bez kojih se ne može), nakon čega opet dobijate klasični upit da li želite da se novostvoreni imenik NWLITE automatski uključi pod komandu **path** u vašoj autoexec.bat datoteci.

Zatim sledi pitanje da li se slažete da se komanda **lastdrive** (redno poslednji disk u sistemu) ugradi u vaš autoexec.bat, iako imate opciju da ili ne, savetujemo potvrdan odgovor za mreže do 8 mašina (standardno, DOS priznaje 5 disk jedinica, a ovim se broj proširuje za daljih 9 – malo rezerve nije na odmet). Na kraju ćete videti pozdravni meni.



Pa da počnemo...

Nakon uspešno obavljene instalacije, možete početi. Ako ste je vi za vaše potrebe instalirali, mreža vam (kao i sve ostale "velike" mreže) dozvoljava da budete rukovodilac, ili mali (lokalni) Bog. Kada se kao takav ubacite u red programa, pozdraviće vas meni prikazan na slici 3.



Da se ukratko upoznamo sa ponudom. Prva opcija ima dve podopcije: prijem poruka (**Receive messages**) i slanje poruka (**Send messages**). Obe možete uključiti (on) ili isključiti (off). Vrlo zgodno ako vam je (i) supruga na mreži: onda lepo sve oba u "off" i najzad steknete malo mira. Divni su ti računari – pretvorite suprugu u instant udovicu (samo pazite da ne preterate).

Druga opcija takođe ima podmeni sa 11 podopcija, od imena korisnika, određivanja nivoa privilegija, obaveznosti lozinke, njenom trajanju i obnavljanju, itd. ukratko, sve što je potrebno da definišete korisnike od sebe samoga pa na dalje. Međutim, ovde ne možete bilo šta promeniti, ovo je samo informativna slika koja opisuje kako je šta postavljeno, a same izmene morate obaviti na drugim podmenijima u okviru narednih opcija.

Treća opcija **(Map drive letters).** Tu možete videti koliko disk jedinica postoji u celoj mreži, koji imenik na kom disku je tekući radni imenik, koja mašina je server i najzad "pravni status", odnosno sloboda pristupa disku. Ovim poslednjim možete nekome dati ili oduzeti mogućnost pristupa ovoj ili onoj mašini, odnosno imenu na mašini.

Četvrta opcija se odnosi na status štampača. Omogućava vam, na primer, da odredite da li će laserski štampač povezan za vaš računar, koji je trenutno server, biti dostupan celoj mreži ili samo vama. Određivanje se svodi na dovođenje kursora na željenu liniju (LPT1, LPT2 itd) i pritisak na Enter taster, ponavljanjem radnje izvršavate suprotnu naredbu.

Peta opcija vam omogućava da izmenite sopstvenu pristupnu šifru, koja ne sme biti duža od 15 znakova.

JA PA GOSPOD BOG

Najzad, šesta opcija, Supervise the Network ("Nadzor mreže"), je spisak božanskih prava. Na njoj se nalazi čitav niz opcija, obično dostupnih samo sa Olimpa, a ovaj put i u vašem gradu.

Audit log vam omogućava da steknete tačan uvid u događaje u mreži: koliko je uključenja bilo, koliko isključenja, koliko neuspelih pokušaja uključenja (Pera opet zaboravio šifru, grrnblmgrnjwqrtz jedan!), i slično. Te podatke možete zapisati u posebnu datoteku, koju po želji možete i odštampati. Poslednja podopcija omogućava uključenje ili isključenje ove funkcije, pa ako korisnik Pera dnevno tri puta zaboravi ulaznu šifru, uvek možete da dignete ruke od svega i odete u veličanstvenu izolaciju.

Error log pamti koliko puta se na mreži dogodila neka greška, gde i kada. Ako je pomenuti Pera često za računarom, onda ovo možete smatrati i njegovom opcijom.

Network directories omogućava uvid u sve prethodno definisane imenike koje mreža prepoznaje, a opcija **Network printers** u sve štampače koje mreža vidi. Tasterom **Ins** možete dodati nove jedinice, tasterom **Del** ukloniti jedinice sa spiska, a tasterom F3 ih možete preimenovati.

Naredne tri opcije (**Server configuration, Server status** i **Server system files**) opisuju i po želji određuju odnose u mreži. Podsećamo da *Lite* omogućava svakoj mašini u mreži da bude i server i nod, pa otud i potreba za mogućnošću određivanja statusa, odnosno njegove izmene.

Opcija **Time synchronization** ("Sinhronizacija vremena") uprkos tehno-nazivu, zapravo služi ujednačenju časovnika realnog vremena na svim delovima mreže, kako bi beleženje događaja imalo smisla (zamislite da na serveru bude 12 časova, a na nekom nodu 6 uveče).

Najzad, konačna opcija – **Users** ("Korisnici"). Ako je sada već legendarni Pera i dalje na vezi, a vama se već dobro smučio, evo prilike da se izvrši pravda. Vaša moć nad Perom i ostalim korisnicima je ograničena samo vašim shvatanjem morala i humanosti. Možete im izmeniti ulazne šifre, možete im omogućiti ili zabraniti da koriste iste šifre na više mašina, podesiti da povremeno moraju da potvrđuju važnost šifara, odrediti broj dana te izmene, datum kada tekuća šifra ističe i najzad – o božanske li pravde! – doneti Konačnu Presudu, odnosno eliminisati korisnika (funkcija popularno poznata pod nadimkom "Kill user", ili "Ubij korisnika").

MALO PRAKSE – PROGRAMI

Naša lokalna mreža se sastojala od svega dva računara, toliko nam je bilo dostupno na duži period. Napominjemo da *Lite* traži drugačiju kopiju za svaku mašinu, pa smo dodatno bili ograničeni i tom činjenicom – nismo imali obraza da tražimo više.

Dakle, postavili smo dva računara u mrežu i počeli da se igramo: prethodno smo u njih stavili samo programe koji su ili sposobni da rade u mreži, ili su baš namenjeni serverima, primer prvih bi bio *WordPerfect 5.1*, a drugih *WordPerfect Office*. Mi imamo server verziju, koja se od obične, četiri puta jeftinije verzije, razlikuje samo po tome što ima i administratorski podprogram).

Mrežili, mrežili, pa umrežili – radilo je kao bog, bez i jedne jedine greške. Jeste da smo se malo namučili oko programa *WP Office*, ali smo problem locirali u tom programu (to znači da smo zabrljali prilikom instalacije, a pošto smo mi gazde, opet je kriv program). Hardverašima su uvek krivi programeri i obratno, u tome i jeste pola zadovoljstva računarstva – imati dežurne krivce. No, šalu na stranu, nakon ispravljanja naše greške, sve je proradilo kako treba.

Bilo je muke, istina male, oko podešavanja egzotičnijih uređaja. Recimo, omogućavanje pristupa faks/modem kartici se pokazalo kao dosta težak posao. Dobro smo se namučili, da bi se ispostavilo da je došlo do kolizije sa interaptima, odnosno do svađe između dve kartice. Srećom, sve je programabilno, pa smo malo izmenjali interapte i faks/modem kartica je proradila – ali samo lokalno.

Interesantno je gledati šta se događa kada istovremeno poželite da pristupite istoj stvari sa dva računara. To je, naravno, moguće, ali dolazi do primetnog usporavanja. Na dve mašine, od kojih je jedna ipak 386DX sa kešom i 4 MB sistemske memorije, kad i sa kvalitetnim mrežnim karticama, teško da bi se moglo govoriti o drastičnom padu brzina, ali usporavanje je ipak primetno.

Posebno zadovoljstvo predstavlja pisanje i slanje poruka na mreži, naravno više kao pretpostavljena funkcija na nekoj iole većoj mreži – recimo na dve sprata i u suprotnim delovima zgrade. Funkcija ne samo da dostupna već se i lako koristi, toliko lako da prosto mami čoveka da piše poruke. Pa, ako je koleginica još i zgodna...

Sasvim očekivano, računar u ulozi servera ne možete čak ni slučajno tek tako resetovati sa **Ctrl-Alt-Del** tasterom, dobićete kontrolni meni sa obaveštenjem o broju trenutno aktivnih veza (nodova na mreži) i brojem otvorenih procesa. Biće vam ponuđene dve opcije – ili ćete zatvoriti sve procese bez resetovanja računara ili ćete resetovati računar. Dakle, pamet u glavu, dobro razmislite.

Uzgred, jedna mala glupost koju program pravi prilikom instalacije: pita vas da li želite da sam sebe uključi u spisak imenika (*path*). Vi naravno odgovorite potvrdno a on to uradi, ali na najgluplji mogući način – na kraju vaše autoexec.bat datoteke jednostavno doda ko-

mandu path – c /nwlite i time totalno onesposobi sličan komandni red iznad njega u koma se nalaze podaci o svemu drugome što sta sami prethodno uneli. Naravno rešenje je da obnšeta tej red i dodate isti podatek na kraj postojeće komende, kako ne biste izgubili njenu ukupnu funkcionalnost. Nije neki problem ali ovo napominjenmo da se ne biste iznenedili ako prilikom prvog narednog podizanja sistema zatražite nešto pa dobijete poruku *file not found* ili nešto slično

MALO PRAKSE – MAŠINSKA OPREMA

Pažljiviji čitaoci će se setiti da smo proteklih meseci razvili jednu mašinu namenjenu ulozi servera upravo u ovakvim mrežama. Naravno baš ta platforma nam je i poslužila za svo ovo igranje sa mrežom. Da podsetimo ukratko: radi se o platformi sa procesorom 80386DX na 33 MHz i 64 KB keš memonje, 4 MB sistemske memonje tvrdim diskom Seagate ST3144 (130 MB) i jednom WD 90C30 VGA karticom sa svim pratećim sadržajima. Sva u svemu jedna prilično standardna konfiguracija, istina sa dosta uloženog rada u što bolju integraciju sistema.

Ovakva kakva je, ta konfiguracija u celosti zadovoljava sve potrebe održavanja manje mraža sa 5-6 korisnika. Sigurni smo da bi se u tom slučaju i te kako osetilo usporavanje rada diska koji je sasvim prosečan uređaj koga ionako ne krasi neka posebna brzina a kapacitet mu je diskutabilan za ozbiljniji rad

Što se kapaciteta tiče, to ćete lako rešiti zamenom za disk većeg kapaciteta. Što se pak brzine tiče, zamenom standardnog IDE kontrolera inteligantnim kontrolerom sa keš memorijom, dobili smo efekat maltene nultog čekanja racionalno znamo da je do nekakvog usporavanja moralo doći ali je bilo toliko malo da ga zaista nismo primetili. Terajući ovu logiku do kraja, na kontroler smo stavili samo jedan SIM modul sa 1 MB, čime je svaki korisnik dobio polovinu te sume, odnosno po 512 KB. Čak i ova količina, uz sve uključene funkcije kasnijeg zapisa (kašnjenja od 15 sekundi, maksimalna dužina keširanog reda, maksimum unapred učitanih redova, itd.) se pokazala više nego dovoljnom za veoma udoban rad.

Ekstrapolacijom zaključujemo da bi disk od nekih 200 MB sa ovim kontrolerom (DC-600) i, recimo, 4 MB keš memonje bez ikakvih problema i uz apsolutno minimalnu degradaciju brzine, sasvim lako i lepo (a to znači brzo) opsluživao 8 korisnika. Ovaj broj omogućava i po 12,5 MB prostora za podatke svakog korisnika ponaosob, uz pretpostavku da je polovina diska zauzeta raznim programima.

Što se sistamske memorije tiče, 4 MB u principu zadovoljava 4-5 korisnika, ista količina memorije će zadovoljiti i više korisnika, ali uz progresivnu degradaciju performansi. Kao neko opšte pravilo, računajte na 512 KB sistemske memorije po korisniku. Logikom apsolutnih kapaciteta, proizilazi da bi ovakva platforma mogla da podrži do 64 korisnika što uz dodatu po 256 KB kapaciteta na keš kontroleru u celini zadovoljava sva zahteve

MALO EKONOMIJE I LOGIKE

Novell Lite je odličan program za mala mreže, pri čemu se pod malim podrazumevaju mreža do dvadesetak korisnika. Veoma je ekonomičan po prostoru koji zauzima na disku (je dva 430 KB sve u svemu), jeftin po jednom čvoru, fleksibilan (omogućava laku nadogradnju odnosno dodavanja) i, koliko smo mi imali prilike da vidimo, izuzetno je otporan na greške. Ovo smatramo sigurno najvažnijim elementom – jer džaba nam sve ako mreža svaki čas pada

Pošto su mu zahtevi za mašinskom opremom skromni, potencijalni korisnici zaista imaju slobodu izbora, u principu sve od SX na 25 MHz bez keša i više je prihvatljivo, a kriterijum je prosta brzina rada

Mogućnost lakog određivanja i jednako lake izmena statusa svake mašine (svaka može biti server, nod ili i server i nod) otvaraju vrata gotovo neograničenoj komunikaciji, kako po obimu tako i po kvalitetu unutar mreže kao celine. Ovde nema straha od nadogradnje

Brzina rada je problem o koma treba razmisliti. Sve zavisi od toga čime se bavite i lično i na mreži. Što su poslovi sa obe strane složeniji očekujte veće padova brzine. Dalje dobro razmislite da li želite da vaš server bude samo server ili i server i nod: jasno je da posvećeni server (koji dakle ne može biti i nod) radi brže od neposvećenog. Ako rešite da jednu mašinu definišete isključivo kao server, razmislite i o inteligentnom keš kontroleru – verovatno bi vam dobro došao, a možda vam je čak i neophodan

I najzad, malo ekonomije. Razmotrimo tri slučaja mreže: sa 3 korisnika, sa 6 korisnika i sa 12 korisnika. Pralom cena u DEM dajamo u sledećoj tabeli

Mreža	3 korisnika	6 korisnika	12 korisnika
Oprema			
80386DX-33	1 200	1 200	1 200
disk 12 MB/korisnik×2	700	1 300	2 600
RAM 512 KB/korisnik	150	300	1 020
keš disk kontroler	-	-	650
256 KB/korisnik	-	-	270
VGA mono	350	350	350
Nod (AT 286, 1 flopi 1 MB RAM Hercules tastatura, I/O	2 700	5 400	10 800
Mrežna kartice	1 050	2 100	4 200
Štampač (matrični 9 iglica 1 na 3 korisnika)	600	1 200	2 400
Novell Lite	900	1 800	3 600
Cena sa Novell Lite	6 950	13 650	27 090

PROFIL: MICRO AIR /ANIMA

ELEKTRONSKI PROJEKTNI BIRO

Našim redovnim čitaocima su, bez sumnje, dobro poznata imena MICRO AIR i ANIMA. Ovu priliku koristimo da pobliže prikažemo ova dva srodna preduzeća, koja u svojim poslovima koriste najprefinjenije informacione tehnologije.



U poslednjih nekoliko godina sve je jasnija neophodnost računarske automatizacije poslova vezanih za sve vrste projektovanja. Postupak kojim se vrši automatizovanje ovih poslova je CAD (Computer Aided Design), a uz njega su usko vezani 3-D modeliranje, fotorealistična vizualizacija i animacija, kao i multimedia veza sa audio i prevashodno video svetom

ANIMA je preduzeće za kompjutersko arhitektonsko projektovanje, vizualizaciju i animaciju. Osnivala su ga trojica arhitekata, želeći da udruže svoje individualne napore i znanje na polju primene računara u arhitektonskoj projektantskoj praksi

Delatnosti preduzeća ANIMA obuhvataju sve uobičajene vidove rada arhitektonskog projektnog biroa. Ali zahvaljujući primeni savremene CAD računarske tehnologije, tu su i nove, veoma atraktivne mogućnosti izrade raznih vrsta arhitektonskih predstava kroz fotorealistične prikaze i animacije. Osim pomenutih usluga ANIMA svojim poslovnim partnerima nudi i konsalting iz oblasti nabavke i primene savremene računarske opreme u arhitektonskoj praksi, kao i obuku kadrova u okviru *AutoCAD* i *CorelDRAW* škola. Kursevi se obavljaju na najkvalitetnijoj računarskoj opremi velikog kapaciteta uz visokostručne predavače tako da se polaznici lako i brzo upoznaju sa svim mogućnostima softverskih paketa

Za sve one koji bi želeli da koriste mogućnosti računarski podržanog projektovanja a nemaju neophodnu opremu, ANIMA nudi iznajmljivanje visokokvalitetnog HP plotera, laserskog i kolor štampača, kolor skenera i slajd rekordera. Ako imate nameru da svoje računarski generisane grafičke radove prezentirate svojim poslovnim partnerima, sliku sa računara možete preneti na video traku, što je idealan medijum za sve vrste prezentacija

Na arhitektonskom polju ANIMA je takođe izuzetno uspešna firma sa zapaženim i nagrađenim učešćem na velikom broju značajnih konkursa. Jedan od značajnih vidova kooperacije je saradnja sa arhitektima na velikim projektima pri čemu računarska automatizacija projektovanja u potpunosti dolazi do izražaja.

MICRO AIR i ANIMA obrazuju zatvoren ciklus, s tim da je glavna delatnost MICRO AIR-a prodaja specijalizovanih high-end PC CAD kao i grafičkih računarskih konfiguracija. Pored brzih 386 i 486 računara, prisutan je i asortiman HP, Hercules, Diamond i EIZO proizvoda vezanih za oblast CAD i grafike

MICRO AIR ima veliki broj referenci na CAD i grafičkom polju kako u Beogradu tako i u unutrašnjosti. Oprema koju nude potiče od kvalitetnih i pouzdanih proizvođača i ima vrhunske performanse. Jedan od zanimljivih novih trendova su i CAD mreže koje daju dodatni podsticaj produktivnosti

MICRO AIR i ANIMA otvoreni su za sve vrste saradnje u ovoj oblasti. Za dodatne informacije obratite se na adresu 11000 Beograd, 28. Novembra 71, tel 011/343-940 *(N. N.)*



zašto ne
ODMAH
Panasonic
TELEFONI CENTRALE
TELEFAXI
GENERAL ELECTRONICS
Genel
21000 Novi Sad; Trg Dositeja Obradovića 7
Telefon: 021/350-444 Telefax: 021/350-999

Only Oracle CASE allows teams of developers to jointly design and build database applications that run on virtually all the computers in your organization. Mainframes, minicomputers, workstations and PCs.

Oracle CASE supports the full lifecycle of systems development. From strategic planning, analysis and design to on-line generation, production and maintenance. Using sophisticated diagrammers for entity-relationship models, function hierarchies, dataflow analysis and matrices. All integrated with a comprehensive set of application development tools and utilities.

Oracle CASE lets you develop applications anywhere. And run them everywhere.

Developers' efforts are fully coordinated via a shared, on-line repository. Which can reside on virtually any platform, and be accessed by just about any combination of terminals and workstations. So developers can truly work as teams to improve productivity, while eliminating errors and redundancies.



And once CASE generates your application, you can run it on virtually all your computers. From PCs through mainframes. Oracle also offers comprehensive services to transfer our CASE expertise to you. Including full support, education and consulting to maximize your success with CASE technology.

ORACLE

Software that runs on all your computers.

tel 011/2224748 fax 011/2223052

BEZ PO MUKE

Da su fontovi jedna od stvari koja je još dalako od konačnog rešenja imao je prilike da se uveri svako ko je na bilo koji način došao u dodir sa njima. Nabrojeni standardi za format zapisa fonta i ogromna količina informacija (kod bit-mapiranih), odnoano komplikovanost i nedokumentovanost zapisa (kod vektorski opisanih fontova) otežavaju razvoj slatki za ovu vrsta posla, pa se broj programa može izbrojati na prstima jedne ruka. Ako vam u poslu nedostaje ova vrsta softvera, za vaa imamo jednu radosnu vast.

Zoran Životić

Tek od skora u oblasti fontova stvari su počele da se odvijaju u boljem pravcu – prvo smo dobili već gotovo rešenje latiničnih pisama sa našim slovima (Bitstream), potom je definitivno presuđeno u sporu bit-mapa prema vektorskoj definiciji u korist ove druge, pa je i problem ogromnog disk prostora skinut sa dnevnog reda. Program koji je pred nama, TIPOTEK, pretenduje da konačno reši i poslednji problem vezan za fontove – izradu potpuno novog fonta, uz mogućnost rada sa svim trenutno važnim vektorskim formatima.

GVOZDENA ZAŠTITA

Program se isporučuje na jednoj 1.44 disketi uz uputstvo od oko 80 strana A5 formata. Instalacija se izvodi veoma jednostavno – po startovanju programa INST, treba u priloženu masku upisati osnovne podatke (pored direktorija u koji želite da instalirate program tu su i registracione informacije – naziv vlasnika, adresa, itd). Instalacija se završava za nekoliko minuta i na disku dobijate samo jedan direktorij u kome su sve potrebne sistemske datoteke – ukupno nešto manje od 900K, uz nekoliko primera fontova i par skeniranih predložaka slova.

Program ima ugrađenu zaštitu – nakon instalacije na jednoj mašini može se ponovo instalirati samo na toj istoj mašini, a sama instalacija može isključivo na toj mašini da radi. S obzirom da se ovakva vrsta zaštite bazira bilo na principu „neregularnog" zapisa na disk bilo na uzimanju neke vrste „otiska prsta" računara (kombinuju se razni parametri – od tipa biosa i diska do čitanja serijskog broja IDE diska) mogu se zamisliti situacije u kojima biste mogli imati određenih problema sa ovakvom zaštitom čak i ako ste sasvim legalno registrovan korisnik. Ipak, verujem da ćete, nakon što pošaljete datoteku TIPOTEK.USR i postanete registrovani korisnik, imati punu pomoć od firme i u ovakvim retkim spornim situacijama.

TIPOTEK je veoma raspoložen prema najrazličitijim konfiguracijama računara. Potrebna je u osnovi čista DOS konfiguracija sa 640K i jedan od nekoliko podržanih video sistema – EGA, standardna VGA sa rezolucijom 640x480, VGA sa rezolucijom 800x600 ili 1024x768, hercules i pomalo zaboravljeni ali za ovu namenu izvanredni WYSE 700. Program će sam prepoznati adapter sa kojim radi, ali je obezbeđen i prekidač kojim se forsira određeni režim pri startovanju programa.

Program u potpunosti omogućava rad sa tastature, ali se ipak pun komfor dobija tek sa mišem, koji mora da ima svoj drajver (MOUSE.COM ili šta već uz miša koristite).

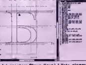
WINDOWS NA DOS NAČIN

Program TIPOTEK predstavlja, u osnovi, font editor u najklasičnijem smislu – učitate font i zatim interaktivno menjate izgled slova, dodajete nove znake, vršite neku lokalnu ili globalnu transformaciju slova (rotiranje, naginjanje i slično) ili premapiranje znakova u fontu. Ono što ga čini retkim u njegovoj kategoriji je činjenica da se bavi do skora nedodirljivim vektorskim formatima zapisa fonta – tu je pre svega, *Adobe Type 1* (PBF), zatim *LaserMaster* (LXO), *CorelDraw* (WFN), *Zsoft Publisher"s Type Foundary* (OTL) i URW. U pripremi je i novi *Windows* standard *True Type* (TTF). Kao izlazni format, mogu se dobiti i čist bit-mapirani font u SFP formatu u bilo kojoj veličini između 4 i 72 pt ili *Adobe Type 3* (PFA), u čitljivom PostScript obliku.

Ono što će vas verovatno iznenaditi pri prvom susretu sa programom je da je grafički interfejs potpuno originalan – umesto *Windows* aplikacije kakvu biste verovatno očekivali, dobijate potpuno samostalan program, koji sve svoje elemente grafičkog interfejsa kreira sam. U principu sam protiv ovakvih rešenja – grafički interfejs uspostavlja prilično složenu interakciju sa korisnikom (broj mogućih radnji je dosta veliki iako tako ne deluje na prvi pogled), pa je za privikavanje potrebno određeno vreme. *Windows* je ovaj zadatak već obavio nametnuvši se kao apsolutni standard, pa deluje prilično neracionalno smišljati nešto novo. Ipak, TIPOTEK predstavlja malo iznenađenja – neće vam biti potrebna gotovo ni jedna jedina sekunda za navikavanje, jer je ceo skup radnji sveden na minimum (nema prozora, od klasičnih elemenata tu su samo skrol kontrola, liste, ikone i OK i CANCEL dugmad) i izveden na standardan način. Brže izvođenje komandi sa tastature će zahtevati navikavanje, ali je rad sa mišem potpuno intuitivan i saglasan *Windows* iskustvu.

GRUBO CRTANJE...

Kada se kaže „program omogućava editovanje fontova" onda se iz takvog opisa teško



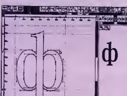
Radni prostor: Slovo (levo) i lista njegovih segmenata (desno)

može nazreti sva složenost ovog posla i naslutiti šta sve program može da uradi. Zato ću ukratko opisati faze u procesu kreiranja fonta, kako bih mogao da pokažem kako se program nosi sa svakom od njih.

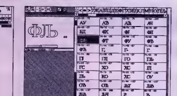
Poći ću od najgore pretpostavke da želite da napravite potpuno nov font. Prvi problem sa kojim ćete se srasti je pitanje uzorka – dizajn slova je posebna i izuzetno složena disciplina kojom se ne možete baviti usput. Jedino što vam ostaje je da krenete sa nekim uzorkom – otiskom slova sa nekog foto-slog uređaja u malo većoj dimenziji i da potom kreirate font po njemu. Umesto ručnog crtanja rastera po vašem uzorku, program TIPOTEK nudi logičniju i jedinu moguću konformu opciju – uzorak možete skenirati (u PCX formatu) i potom ovu sliku postaviti na ekran kao mustru preko koje definišete znak. Već na ovom prvom koraku ćete uočiti da pred sobom imate ozbiljan program koji vodi računa o najsitnijim detaljima. Ako biste skenirani otisak slova samo postavili na ekran, pojavio bi se problem preklapanja linija koje sami crtate sa crnom površinom slova. Zato program omogućava prikaz uzorka u horizontalnom, vertikalnom ili tačkastom rasteru, ili u konturnoj formi. Uzorak se može povećati ili smanjiti prema potrebi i jednim potezom miša postavljati ili uklanjati sa pozadine.

Sada je na redu opisivanje konture slova nizom segmenata – Bezijerovih krivih i pravih. Crtanje Bezierovih krivih je najdelikatniji deo dizajna slova, jer se menjanjem položaja kontrolnih tačaka utiče na oblik segmenta koji treba maksimalno poklopiti sa uzorkom. Ako se uzme da je najčešći slučaj da jedna Bezijerova kriva pokriva prilično velike segmente (najčešće deo kod koga se ne menja smer savijanja, na primer jedna četvrtina oboda slova O), onda je jasno da nije uvek lako menjanjem položaja kontrolnih tačaka poklopiti celu konturu. Zato program, umesto potpune kontrole u prvom crtanju, nudi skraćenja postupka koje se bazira na određenim preporukama dobrog dizajna – sam određuje početni položaj kontrolnih tačaka, pri čemu koristi normalno postavljene tangente, od kojih je jedna uvek horizontalna.

Efekat ovog skraćenja postupka je izvanredan, jer se na lak način pokriva najveći broj situacija, a eliminišu se i manja lutanja u dizajnu, u koja bi vas mogla odvesti nesavršenost otiska koji koristite kao uzorak. Naravno, položaj svake kontrolne tačke može se naknadno promeniti i to biranjem više režima – jedan, u kome zadržavate poklapanje obe tangente u dodirnoj tački dva segmenta (kontinualan spoj) i drugi, o kome nezavisno menjate položaj svake pojedinačne kontrolne tačke (oštar spoj). Dodatnu kontrolu omogućuje niz opcija kojima se dalje vrši izmena segmenata – pomeranje, promena veličine i razmera, i estetski posebno simpatična opcija (ikona u obliku noža) za sečenje segmenta na više delova. Sliku upotpu-



Radni prostor: Slovo "f" sa istaknutim kontrolnim tačkama



Kern tabela: Interaktivno podešavanje razmaka između slova

njuju i detalji, kao što su opcije za fiksiranje x ili y koordinata kako bi se izvelo besprekorno horizontalno ili vertikalno pomeranje, kao i, konačno, potpun *clipboard* mehanizam kojim se mogu prenositi delovi ili celi crteži znaka. Najkraće rečeno – ukoliko uz sve mehanizme koje program nudi imate i dalje probleme sa formiranjem kontura znaka, razlog može biti samo neiskustvo – kada steknete malo prakse, shvatićete koliko brz rad omogućavaju ovi mehanizmi.

U toku opisivanja znaka, na ekranu imate sve relevantne informacije – od trenutne pozicije kursora (koji može biti u obliku pokazivača ili ukrštenih osa), preko položaja svih kontrolnih tačaka, do signalizacije otvorenih segmenata i prikaza konačnog izgleda trenutnog dizajna slova u odvojenom delu ekrana. Umesto grafičkog prikaza možete u desnom delu ekrana dobiti i tekstualni prikaz – listu svih segmenata sa njihovim parametrima. Ovaj listing je jedna od retkih stvari u programu koja ima neiskorišćen potencijal. U nekim situacijama bi mogućnost selekcije segmenta iz ove liste bila daleko jednostavnija od varijante pronalaženja na crtežu. Lista bi se, zapravo, mogla upotrebiti kao alternativni način pozicioniranja na određeni segment i selekcije grupe segmenata koji čine zatvorenu konturu (na listingu su već ionako grupisani).

...I FINO DOTERIVANJE

Kada opišete znak, vaš posao nije ni iz bliza gotov. Sledeća delikatna operacija je dodavanja informacija koja pomažu programu za generisanje raster forme slova da ovaj posao uradi što je moguće korektnije i pri niskim rezolucijama. Ako se kao tipičan primer uzme veliko latinično slovo H, i ako se zna da pri veličini slova od, recimo, 8 pt na 300 dpi, stub slove H može biti širok samo 2 ili 3 tačke, jasno je da se kod proračuna može desiti (zbog zaokruživanja i „nezgodnog" polaganja slova na raster) da stubovi budu nejednake širine. Na pomenutim veličinama slova, ovakav rezultat deluje katastrofalno, pa su zbog toga uvedene takozvana hint-informacija u font. Na konkretnom primeru, to znači da na crtežu slova treba obeležiti šta čini trup slova, pa će se rasterizator postarati da ovi trupovi budu jednaka širine. TIPOTEK u potpunosti podržava ovaj mehanizam, ali ne i sve ostale hint- informacija koja je definisao *Adobe*. U svakom slučaju, širina trupa je najvažnija od njih i ima najviše uticaja na konačan izgled slova.

Još uvek niste prevalili ceo put. Sledeće na listi je pitanje širina slova – potrebno je postaviti takozvanu referentnu tačku i pomak koji referentna tačka treba da učini nakon što se slovo odštampa. TIPOTEK i ovde nudi zgodno rešenje – sa obe strane slova možete postaviti isti ili različite znake i tako, interaktivnim menjanjem širine, pratiti efekat. Obično se u okolinu postavljaju slova sa oblom, ravnom i zakošenom stranom (O, A i H su dobar izbor). Podešavanje širina je prilično mukotrpan process pa bi ipak bilo zgodnije da program za ovu namenu ima poseban režim, u kome bi tekuće slovo bilo prikazano istovremeno u svim ovim okruženjima.

Ovde bi posao mogao biti priveden kraju, ali ne i ako želite da font dovedete do perfekcije. Za ovo je potrebna dodatna informacija – takozvano podsecanje parova slova (kerning). Slovo A i V, svaki sa korektno postavljenom pojedinačnom širinom, u međusobnom sudaru daju izrazito loš efekat. Zato se za ovaj par uvodi podsecanje – dodatno smanjenja (ili povećanja) razmaka kada se ova dva znaka nađu jedan porad drugog. Program za ovu namenu ima poseban režim – na ekranu dobijete tabelu svih „nezgodnih" parova i u desnom uglu odnos trenutno izabranog para. Jednostavnim pomeranjem postavljate kern vrednost.

Nakon ovoga, ostaje vam još samo da font finalno snimite u izabranom formatu. Dok razvijate font, možete koristi interni format programa (IFT) koji program najbrže obrađuje. Program je potpuno transparentan što se formata tiče – učitani *Adobe Type 1* možete snimiti kao *Corel WFN* i obratno. Za one sa HP kompatibilnim laserima koji konsta bit-mapirane fontove, posebno je interesanta opcija za direktno generisanja SFP fontova – upišete veličina koja su vam potrebne i program ih samostalno generiše. Brzina je veća od *Bitstream*-a koji radi taj isti posao – na 386/33Mhz mašini bez koprocesora, font sa 7-bitnim skupom znakova u veličini od 12pt se generiše za oko 30-tak sekundi.

U toku rada sa programom na raspolaganju su vam i dve grupe HELP tekstova – jedna je klasična (23 ekranske strane) sa najkraćim spiskom opcija, skraćenica (tastera) i opise opcija. Druga grupa je nazvana *Obuka* (33 strana) i daje širi opis pojedinih operacija na sličan način kao i uputstvo. Ukoliko imete iskustva i znate šta znači dizajniranje fonta, gotovo da vam i neće biti potrebno uputstvo – sve opcije se zaista intuitivno pronalaze preko više nego slikovitih ikona. Za one manje iskusne, uputstvo prepuno ilustracija je više nego dovoljno da se brzo upoznaju sa programom, pa čak i sa nekim detaljima vezanim za sam dizajn slova. Ipak, ukupan utisak koji se nameće, ne samo na osnovu uputstva, jeste da se pre svega radi o programu namenjenom profesionalcima (šta god da to u našim uslovima znači).

PROGRAM SVETSKE KLASE

Verujem da mnogi od vas, osim što im je naziv programa zazvučao srpski, nisu do sada zaključili da se radi o programu domaće izrade. TIPOTEK je program beogradske firme INFOTEK i, bar po copyright poruci, isključivo delo jednog čoveka – **Milena Velimirovića** (*mivel* na Sezamu). I zaista, ovde ćete pronaći malo šta od onoga čemu smo obično spremni da progledamo kroz prste kada je u pitanju domaći proizvod – ako bi se uputstvo i moglo smatrati neatraktivnim i prekratkim za svetske standarde, sam program je po mnogo čemu proizvod svetske klase. Moja jedina generalna primedba, da bih više voleo da vidim ovakav program u *Windows* okruženju (program bi se time rašio i nekih manjih problema sa nedostatkom memorije i potrebom da „juri" sve režime i tipove VGA karti), potpuno je nevažna u kontekstu upotrebe programa. INFOTEK radi brzo, njime se lako upravlja i poseduje skup izuzetno promišljenih komandi i opcija za najraznovrsnija poslova sa slovima - od lakih ili složenih intervencija u postojećim fontovima do stvaranja potpuno novih pisama - koje su očigledno posledica dugog iskustva autora u radu sa fontovima.

Kod nas je, kada su fontovi u pitanju, ostalo još dosta toga da se uradi (fond ćiriličnih fontova je više nego skroman) – uz ovaj alat za takvo stanje više neće biti opravdanja.

Korisna adresa

INFOTEK
11000 Beograd, Dunavski kej 21
Tel 631-547

ELEKTRONSKI HRAM ŠAHA

Na tržištu postoji mnogo programa za rad sa bazama podataka. Ni broj programa za igranja šaha nije mali. Šta li bi onda trebalo da znači „baza podataka za šah"? Mnogi od vas već pogađaju o čemu se radi – to su, jednostavno rečeno, programi prilagođeni za rad sa velikim brojem šahovskih partija. To bi trabalo da olakša rad igračima-profesionalcima, šahovskim novinarima i piscima-teoretičarima, ali, zašto da ne, i masi amatera.

Zoran D. Brkić

Vaš pisac je imao priliku pre nekih petnaestak godina da uđe u najtajniji hram sovjetske šahovske kulture u biblioteku Centralnog šahovskog kluba u Moskvi. Ono što me je najviše zadivilo nisu bili primerci najređih knjiga, već kartoteka od stotina hiljada uredno ispisanih kartica sa šahovskim partijama, poput onih kakve koriste bibliotekari. Prvo što mi je palo na pamet bilo je koliki je tu ljudski rad utrošen, a odmah zatim kako se tu snaći. Moji domaćini su me impresionirali velikom veštinom – začas su pronašli u toj gomili čak i jednu moju partiju koju ni sam nisam imao! U to vreme Botvinik je već postizao uspehe sa svojim programom za igranje šaha, te je pravo čudo kako se Rusi nisu prvi dosetili da primene informatiku i u manje atraktivnim aspektima šahovske igre. Sigurno je svoju ulogu odigrala i pojava PC računara. Tu su Rusi zaostali, pa su se prvi programi – šahovske baze – pojavili na Zapadu. Danas ih već ima nekoliko: holandski „NICBASE", nemački „Chess Base", te ruski „Paragraf", „Pat" i „Dialog".

ŠIFARSKI SISTEM

Ovog puta govorićemo o holandskom „NicBASE"-u, koji spada u najstarije te vrste, a verovatno i najrasprostranjenija programe. On zaslužuje apsolutni primat, videće se i zašto. Počelo je tako što je holandska firma „New In Chess" pokrenula pre desetak godina šahovski časopis, u obliku debele polugodišnje knjige sa nekoliko stotina partija. Ideja je isuviše podsećala na naš „Šahovski informator", pa je čak došlo i do spora. Glavni kamen spoticanja je bio šifarski sistem. Srž tog sistema je da se partije klasifikuju prema otvaranjima u nekoliko glavnih grupa, koje se zatim označe kombinacijom slova i brojeva. Bilo je to neophodno jer se šahovska teorija toliko razvila da uobičajeni nazivi otvaranja, a potom i varijanti, nisu mogli obuhvatiti stotine novoustanovljenih podvarijanti. „Informator" je napravio grubu podelu prema prvom potezu, na E (1 e4), D (1 d4) i R (sve ostalo). Zatim se iza slova dodaju silne cifre koje označavaju otvaranje i varijantu.

Mnogi profesionalci – a to su obično ljudi s natprosečnom memorijom – su se žalili da ljudski mozak jednostavno ne može sve to popamtiti! Holanđani su našu ideju usavršili i dali rešenja mnogo jednostavnija i logičnija. Oni su iskoristili dva slova da označe naziv svakog od važnijih otvaranja. Tako je Sicilijanka (Najdorfova varijanta) „Si 4" umesto „E 94", Francuska odbrana (varijanta 3 e5) „FR 3", umesto „E 00", Kraljeva indijka „KI 1-80", umesto „D 71-99", itd. Tako se uspostavlja izvestan stepen asocijacije, što znatno pomaže. Kad su se pojavili PC računari, naš „Informator" je započeo saradnju sa Nemcima, koji su razvili program „Chess Base", konsteći pomenuti rogobatni šifarski sistem. S druge strane, Holanđani su taj posao uradili sami. Oni su imali dobre tradicije i u programiranju (poodavno je IBM locirao svoju centralu u Amsterdamu) i u šahu (njihov Dr. Euwe je bio pre rata svetski prvak).

U NISKOJ REZOLUCIJI

Program stiže na sedam disketa. Tri diskete zovu su *Program*, *Data* i *Tools*, a na preostale četiri je *Nickay* - šifarski sistem. Hardverski prohtevi su minimalni. Jedini je uslov da imate miša – bez njega program neće raditi. Instalacija se obavlja tako što se startuje odgovarajući program sa *Tools* diskete, no moguća je i „ručna" instalacija. *NICBASE* nema nikakve zaštite, ali zato dobijate verziju sa već upisanim vašim serijskim brojem i imenom i prezimenom! Postoji i indirektna zaštita. Svaka nova verzija programa ima svoj format datoteka, te su vam stare datoteke neupotrebljive bez programa za konverziju – kojeg dobijaju samo ranije registrovani korisnici. Novajlije moraju kupovati gotove datoteke od proizvođača. Naravno, sve je to moguće nabaviti od prijatelje, ali je autorima očigledno bila poznata glavna osobina šahista – da vole da kriju jedan od drugog svoje beleške, zbirke partija i slično, kao zmija noge...

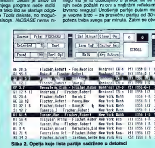
Kad startujete program (NICBASE.EXE), ukazuje vam se (na žalost, u samo dve boje) šahovska tabla u levom delu ekrana i prozor s informacijama s desne strane. Ta slika je i najveća mana programa – grafičko rešenje ekrana je vrlo siromašno. Jasno je da program takve namene ne treba da bude preterano „šaren" kao *Chessmaster 3000*, ali bi ipak trebalo da makar rezolucijom nadmaši ekran programa *Psion Chess*. Verovatno se računalo da kompjuterski slabo obrazovan šahovski narod ne treba opterećivati još i dodatnim upoznavanjem s raznim video-karticama! Prozor sadrže polja sa atributima relevantnim za datu partiju (imena igrača, titule, rejting, mesto, godina i slično), te zapis toka partije u algebarskoj notaciji.

DATOTEKA PARTIJA

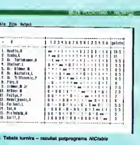
Na vrhu ekrana, pozicioniranjem pointera miša, aktivirate *pull-down* menije. Preko njih ćete komunicirati a programom mnogo češće nego što je to uobičajeno u praksi, ali se na njih neće požaliti ni oni s najbržim refleksima. Izvrsno reaguju! Unošenje partija putem miša je veoma brzo – za prosečnu partiju od 30-40 poteza treba svega par minuta. Zatim se otvori



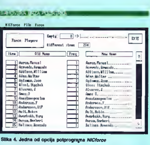
Slika 1. Glavni ekran programa



Slika 2. Opcija koja lista partije sadržane u datoteci



Slika 3. Tabela turnira – rezultat potprograma *NIClable*



Slika 4. Jedna od opcija potprograma *NICforce*

Gamefile i u nju anmi unesena partija. *NiCBASE* tom prilikom pravi dva datoteke sa proširenjima GAM i IND. **Gamefile** sadrži partije u komprimiranom obliku. Tako datoteka koja sadrži sve Fišerove partije (dobijete je na **Data** disku), a ima ih preko 700, ima dužinu 78 k. Za komparaciju neka vam posluži činjenica da postoji knjiga sa istim sadržajem. Ona ima oko 500 strana.

Kada napravite neku svoju datoteku partija, program vam omogućava razne manipulacije njom. Najzanimljivija je opcija **Classify.** Ona će naterati program da svakoj partiji dodali **Key** (kod za klasifikaciju unutar šifarskog sistema). Zatim sortirate sve partije po ključu (odnosno po otvaranjima), te izdvojite samo ona otvaranja koja vas zanimaju. U šahu se često do iste pozicije dolazi raznim redosledima poteza. Program je dovoljno inteligentan da se ima ne da zbuniti. On pamti sve teoretske pozicije putem svog šifarskog sistema i to se moglo očekivati. Ali, imate i opciju da unesete i neku proizvoljnu poziciju **Set up.** Ona će vam pronaći lako i brzo sve partije u kojima se ta pozicija susreće. Što je još lepše, ako vas zanima samo deo šahovske table, na primer određeni strateški tip pozicije ili određena konfiguracija pešaka, izabraćete opciju **Define Pattern,** namestiti na tabli ono što želite i dobićete **Find** - sve partije u kojima se pojavila slična tema, premda se po svemu ostalom one mogu puno razlikovati!

Postoji i druga vrsta datoteka koja se može otvoriti. To je **Database file**, s kojim vam se otvaraju dodatne mogućnosti za rad s ključevima. Usput nastaje još nekoliko datoteka sa proširenjima DBS, POS, PTR. Oni sadrže informacije o **Database** datotekama, kao što su struktura ključeva, pozicije i slično. Ipak su vam najvažniji GAM i IND datoteka. Iz njih se, u slučaju greška, sve ostalo može ponovo napraviti.

Partije iz dadoteka se mogu reprodukovati na ekranu, ali i „odštampati" u „datoteku u ASCII ili „Word Perfect" formatu! Prethodno se može izabrati jezik na kome će biti nazivi figura, zatim notacija (duža, kraća). Ako vam se neka pozicija dopadne, u prozoru predviđenom za komentar unesete znak „#" i u vašem ASCII fajlu će se pojaviti „dotična pozicija, zapisana u standardnom šahovskom maniru. Svaki uneti komentar pojaviće se u tekstu partije unutar zagrada. Dok partiju unosite, možete uneti kraći (ALT + C), ili duži komentar (ALT + L). Moguće je uneti i analizu, tačnije alternative uz svaki potez. Sa (ALT + 1) otvara se prozor s brojem alternative koju možete izabrati pointerom. Ma koliko se vaša analiza dalje granala, program će vas uvek nepogrešivo vratiti do ishodne pozicije. U svetu bez kompjutera šahista u takvoj prilici obično zaboravi da vrati nekog pešaka na pravo mesto i onda nastaju sati mučenja s neposlojećom pozicijom.

Uz program se ne dobija nikakvo uputstvo, što je razumljivo, jer je dosta lak za upotrebu a uz to ima i izvrstan *Help* koji se poziva standardno, sa „f1".

ŠAHISTI ZA ŠAHISTE

I još... Poseban program NICTOOLS.EXE sadrži razne korisne potprograme. Izdvojili treba *NICforce*, koji omogućava da se odjednom u svaki slog (partiju) unesu polja koja se ponavljaju – mesto ili godina i slično, zatim *NICprint* koji štampa dijagrame na *Epson* i *HP* štampačima. Nažalost, kod poslednjeg ne postoji nikakva veza s ASCII tekstom partije, tako da vam jedino preostaje da dijagram isečete i zalepite gde treba. *NICtwice* traži višestruko upisane slogove te ih uklanja, a *NICreorg* definitivno briše slogove koji su, u suštini, samo obeležani za brisanja. Posebna poslastica je *NICtable*. Kad u njega učitate fajl koji sadrži sve partije sa nekog turnira, on vam momentalno daje propisnu turnirsku tabelu sa svim rezultatima, sa izlazom na ekran ili na štampač. Vredi pomenuti još i *NICconvert* koji konvertuje datoteke iz starih verzija u nove, ali i datoteke iz srodnog programa *Chess Base* u matični format.

Sa šahovske strane ovaj paket pruža zaista izvanredne mogućnosti. To je i razumljivo jer su ga pisali šahisti – majstori H. i N. Kuijf. Program je pisan u jeziku Turbo C. Programski gledano, mora se priznati da krupan posao koji mu predstoji obavlja zaista vrlo brzo. Algoritmi sortiranja i pretraživanja su znatno usavršeni u odnosu na starije verzije. Deklarisano je da program na mašini PC AT 80386, od 33 MHz klasifikuje 320 partija u minuti. Test je rađen na datoteci od 7 000 ključeva, 12 000 pozicija i 60 000 partija. U praksi, utisak je da su rezultati i bolji, ali empirijski nismo provaravali iz čisto praktičnih razloga. Za svoje namene program ne zahteva „svetlosnu brzinu", a nije nam bio pri ruci ni odgovarajući takmac. S druge strane, podatak da je maksimalni mogući broj partija 2 147 483 647 deluje neverovatno – iz prostog razloga što ga je nemoguće proveriti!

Do pre desetak godina, do šahovskog „buma" na Zapadu, vladalo je mišljenje da je do tada u celoj šahovskoj istoriji, otkako su Mavari doneli šah u Evropu, odigrano najviše 70 000 partija. Naravno, onih zapisanih.

Najveći nedostatak programa je već pomenuta grafika. Tu je primećen i jedan uporan „bug" – strelica pointera ne briše se kako valja, pa često dobije brata-blizanca na ekranu! Radi se o već trećoj verziji programa i takvi nedostaci trebalo bi da su otklonjeni.

Firma *New In Chess* izdala je i masu disketa s datotekama. Neke sadrže sva godišta časopisa *New In Chess*, neke su posvećene otvaranjima, čuvenim turnirima ili igračima. Posebno je za aktivne igrače zanimljiv *Quick Disk Service* koji garantuje prijem 20 disketa godišnje sa najaktuelnijim turnirima (oko 120 turnira), sa oko 12 000 partija, ali i po ceni od 450 DEM! No, druge su cene znatno fleksibilnije. U proseku jedna disketa s partijama (oko 500-600 partija) košta oko 30 DEM, ali i znatno manje ako uzimate neki od mnogih kompleta od više disketa.

Treba reći da se sva pomenuto može dobiti i na 3.5" disketama, a za ljubitelje ATARI-ja biće zanimljivo da postoji i, u svemu ravnopravna, ATARI ST verzija programa! U prilog istini, ne bi trebalo izostaviti ni činjenicu da su se Holanđani pokazali izuzetno poslovnim i predusretljivim u komunikaciji s potecijalnim kupcem. S druge strane, Nemci i Rusi ne nalaze za shodno da vam pruže bilo kakve dodatne informacije o svojim programima! Tada valja primeniti stari savet jednog engleskog kompjuterskog časopisa, upućen kupcima softvera: „Firma koja vas ne udostoji ni jedne poštanske marke ne zaslužuje vaš novac"!

Zaključimo da je „NICBASE" izuzetno korisan za profesionalne šahiste. Amaterima i prosečnim kompjuterašima biće dovoljne i osnovna verzija za razonodu. Tada se kao prepreka javlja visoka cena (ravna, na primer, ceni *Turbo Pascala)*, ali kao uteha neka posluži da se ipak radi o programu specijalne namene, i još uvek najjeftinijem od svih sličnih!

Cena: *NICBASE 3.0* 300 DEM, *NIC TOOLS* 195 DEM, *NIC KEY 3.0* 36 DEM.

Kontakt adresa

INTERCHESS BV (New In Chess)
P.O. Box 393, 1800 AJ Alkmaar
The Netherlands
Tel. (0)72-127 137 – Fax: (0)72-158 234


PERFEKCIJOM DO SAVRŠENSTVA
PC-AT 286/16 MHz
Kućište Baby AT
Osnovna ploča 286/16 MHz
Hard disk 45 MB
RAM memorija 1MB
Flopy disk 1.2 MB ili 1.44 MB
Monitor 14" TTL i Herkules karta
Tastatura 101 taster, YU/E
CENA: 1.299 DM
PC-AT 386/40 MHz 64 KB Cache
Kućište Mini Tower
Osnovna ploča 386/40 MHz
Cache 64 KB
Hard disk 45 MB
RAM memorija 4MB
Flopy disk 1.2 MB ili 1.44 MB
Monitor 14" TTL i Herkules karta
Tastatura 101 taster, YU/E
CENA: 1.899 DM
PC-AT 486/33 MHz 256 KB Cache
Kućište Mini Tower
Osnovna ploča 486/33 MHz
Cache 256 KB
Hard disk 45 MB
RAM memorija 4MB
Flopy disk 1.2 MB ili 1.44 MB
Monitor 14" TTL i Herkules karta
Tastatura 101 taster, YU/E
CENA: 2.750 DM
M/B AT 286 16 MHz 182 DM
M/B AT 386 33 MHz Cache 64 KB 567 DM
M/B AT 386 40 MHz Cache 64 KB 584 DM
M/B AT 486 33 MHz Cache 256 KB 1 385 DM
HD 44 MB 22 ms AT-IF Seagate, ST351A 517 DM
HD 85 MB 18 ms AT-IF WD AC280 680 DM
HD 120 MB 10 ms AT-IF Conner CP30104 820 DM
HD 210 MB 15 ms AT-IF WD Caviar 2200 1 280 DM
FD 1.2 MB 5.25 130 DM
FD 1.44 MB 3.5 115 DM
Monitor 14" TTL 220 DM
Monitor 14 VGA mono 270 DM
Monitor 14" VGA kolor (1024 × 768) 730 DM
TVGA 1024 × 768 512 KB 110 DM
VGA Tseng 1024 × 768 1 MB 230 DM
Modem interni 2400 Baud, MNP5 320 DM
Modem externi 2400 Baud MNP5 390 DM
Fax/modem 9600/2400 Bps send/receive 430 DM
Streamer Jumbo 650 DM
Printer Epson LX810 9 pin 180 Cps A4 650 DM
Printer Epson LQ570 24 pin 280 Cps A4 800 DM
LaserJet HPIIIp 3 100 DM
LaserJet HPIII 4 400 DM
Ram karta za LaserJet IIIp/III 1 MB 280 DM
BS PROCESOR
BEOGRAD, Hadži Nikole Živkovića 2
telefon 011/180 739, 181-311 lok. 406
telefax: 011/ 180-739
NIŠ, telefon. 018/51-000, 331-654
NOVI BEČEJ, telefon 023/772 153
mraz
MÜNCHEN Schillerstrasse 22 / III
telefon (99) 49 89 59 59 20
telefax: (99) 49 89 55 35 87

JEZIK ZA SVAKI DAN

Možda ste već postali zasićeni raznim egzotičnim programskim jezicima. Tipična situacija: koristite C, *Pascal* ili nešto slično... i ne pada vam ni na kraj pameti da išta da menjate zbog jedne ili dve dobre osobine nekog jezika za koji niko nije čuo. Ipak, probajte *AWK* – to je nešto vrlo lepo i dopadljivo; veoma neobično i praktično.

Nenad Batoćanin

AWK je mali programski jezik nastao 1977. iz praktičnih razloga. Njegovi autori su vam sigurno poznati: *Aho*, *Weinberger* i *Kernighan* – početna slova su uzeta za naziv jezika. Osnovna namena *AWK*-a je obrada raznih *ASCII* datoteka, odnosno analiza i obrada teksta.

Možda vam se sa ovim neatraktivnim podacima može učiniti da sve to već imate u svom omiljenom programskom jeziku i da vam nešto drugo zaista ne treba. *AWK* i nije napravljen za pisanje velikih aplikacija koje imaju padajuće menije, gomilu opcija itd. On je namenjen poslovima koji svakodnevno niču na vašem računaru i koje je uglavnom potrebno uraditi samo jednom uradili. Za sve što radi *AWK* možete na C-u napisati brži program, ali vam *AWK* (zahvaljujući svojoj koncepciji) omogućava da isti posao obavite za neuporedivo kraće vreme. Možete ga shvatiti i kao svojevrsnu zamenu za makro jezik vašeg editora. U slučaju da vam treba alatka za obradu većih količina teksta, *AWK* bi mogao biti idealan.

DATOTEKE I PROGRAM

Osnovna namena *AWK*-a je rad sa standardnim *ASCII* datotekama. Svaki red datoteke *AWK* vidi kao poseban **slog** izdeljen na **polja** blanko simbolom ili tabom. Cela datoteka je jednostavno niz takvih slogova od kojih svaki sadrži promenljivi broj polja. Ovo je idealno za razne obrade – ne morate sami određivati gde počinje, a gde se završava reč. Datoteka se čita slog po slog, a tekući slog (red) se obeležava sa **$0**. Polja unutar sloga (reči) obeležavaju se sa **$i**, gde je **i** redni broj polja. Na primer, ako je tekući slog

Ovo je recenica

tada je

$1 = "Ovo", $2 = "je", $3 = "recenica",

a ceo slog je

$0 = "Ovo je rečenica"

Pri obradi datoteke postupak je u većini slučajeva isti: otvori se datoteka, redom se čitaju redovi, analiziraju se i na neki način se obrađuju. Zato program pisan u *AWK*-u sadrži samo jednu vrstu konstrukcija: parove uslova i odgovarajućih akcija

```
uslov_1 { akcija_1 }
uslov_2 { akcija_2 }
...
uslov_N { akcija_N }
```

Datoteka se redom čita (slog po slog), za svaki slog se ispituju svi uslovi i izvršavaju se akcije čiji je uslov zadovoljen. Na primer, sledeći program štampa sve redove koji imaju **"PROC"** kao prvu reč:

```
$1 == "PROC" { print $0 }
```

Znači, za svaku učitanu liniju se ispituje da li je prva reč **"PROC"**, a ako jeste, štampa se cela linija. Ako je za isti slog ispunjeno više uslova, izvršavaju se **sva** odgovarajuće akcije.

Ukoliko se ne navede uslov, podrazumeva se da ga zadovoljavaju svi slogovi u datoteci. Sledeći primer ispisuje celu datoteku sa rednim brojem ispred svake linije:

```
{ print NR, $0 }
```

Pošto nema uslova, navedena akcija će se izvršiti za svaki slog datoteke. *AWK* ima nekoliko unapred definisanih promenljivih koje sadrže neke sistemske vrednosti. Promenljiva **NR** je jedna od njih i označava redni broj tekućeg sloga.

USLOVI

Kakvi se sve uslovi mogu zadavati? Za one koji poznaju programski jezik C neće biti nikakvih problema da nauče nekoliko novih operatora – svi standardni C operatori su podržani. Na primer, možete pisati

```
$1 == "DATA" && $3 < 10
```

Međutim, mnogo važnije u radu sa tekstom je to što možete koristiti regularne izraze za specifikaciju uslova. O regularnim izrazima je već bilo reči u "Računarima" – radi se o načinu opisivanja skupa stringova, slično simbolima "*" i "?" u *DOS*-u. Regularni izrazi se obeležavaju simbolima "/". Na primer, sledeći regularan izraz označava string "**proc**" ili "**func**"

```
/proc|func/
```

Naravno, regularni izrazi mogu biti daleko komplikovaniji. Oznaka * iza dela izraza označava nula ili više pojavljivanja, + jedno ili više pojavljivanja, a ? nula ili jedno pojavljivanje. Zagrade [] služe za grupisanje izraza, a **a-b** označava sve simbole između **a** i **b** po ASCII tabeli. Na primer, ceo broj se može predstaviti na sledeći način

```
/[0-9]+/
```

Regularni izrazi služe za proveru nekog uslova pri analizi linije; umesto komplikovanih provera koristi se operator koji se može prevesti kao "sadrži". Na primer, sledeći program ispisuje sve linije koje sadrže broj

```
$0 ~ /[0-9]+/ { print $0 }
```

Ako se regularni izraz zadaje za celu liniju, ne mora se pisati **$0 ~ /.../**, već se može zadati samo regularni izraz. Tako je prethodni program ekvivalentan sa

```
/[0-9]+/ { print $0 }
```

Kao uslov se može zadati i interval: zadaju se dva uslova razdvojena zarezom. Takav uslov ispunjava prva linija koja zadovoljava prvi uslov i sve naredne, zaključno sa redom koji zadovoljava drugi uslov. Na primer, sledeći program štampa sve linije počev od broja 10 zaključno sa linijom koja sadrži string "**STOP**"

```
NR == 10, /STOP/ { print $0 }
```

Postoje i dva specijalna uslova **BEGIN** i **END** koji su zadovoljeni pre početka i na kraju rada sa datotekom. Akcije uz ove uslove se izvršavaju samo jednom i one omogućuju da se neke naredbe izvrše pre (odnosno posle) obrade. Na primer, sledeći program sabira sve brojeve u datoteci i štampa njihov zbir

```
BEGIN { s = 0 }
{
    ls = 0
    for ( i = 1 ; i <= NF ; i++ ) ls += $i
    print ls
    s += ls
}
END { print "Ukupno", s }
```

Na početku se promenljiva s (ukupan zbir) postavlja na nulu. Zatim se za svaki slog računa zbir svih brojeva u tom slogu. Promenljiva **NF** je ugrađena i označava broj polja u tekućem slogu. Obratite pažnju da je za pristup i-tom slogu dovoljno napisati **$i** – nema potrebe ni za kakvom redirekcijom. Za svaki slog se računa i štampa međuzbir, a na kraju se štampa ukupan zbir za sve slogove.

AKCIJE

Za svaki uslov treba navesti akciju (ako se ne navede, podrazumeva se da je akcija **{print $0}**). Već ste videli da je akcija nalik na jedan blok C programa. Može se navesti više naredbi i tada ih treba razdvajati simbolom ";", a svaka naredba se može pisati u više redova. Postoje sve osnovne C kontrolne strukture **if while do for** itd. Sve u svemu, nećete imati problema oko učenja novih naredbi. Pregled dostupnih operatora i funkcija zavisi od implementacije, ali je gotovo sigurno da ćete tu naći sve što vam je potrebno: razni numerički operatori, izdvajanje delova stringa, štampanje itd. Praktično, podržan je gotovo ceo skup C operatora. Jedina razlika je u konkatenaciji stringova, koja se obeležava blanko simbolom

```
a = "abc" "def"
```

U tabeli 1 navedeni su neki važniji operatori i ugrađene funkcije.

Korisnik može definisati svoje funkcije. Definicija počinje sa **function** i ima standardni C oblik

```
BEGIN { s = 0 }
/DATA|VALUE/
{print "Processing Line ", NR , sum($2, $3)}
END { print "Total:", s }
function sum(x, y) {
    if (x > y) s += x
    else s += y
}
```

Ovaj program izdvaja iz datoteke linije koje u sebi sadrže string "DATA" ili "VALUE" i sabire veći od dva broja u toj liniji. Sabiranje obavlja funkcija **sum**.

PROMENLJIVE I NIZOVI

Promenljive u *AWK*-u su veoma neobično rešene. U suštini postoje stringovi i brojevi, ali se ova dva tipa ne razlikuju previše: kada zatreba, izvršiće se konverzija stringa u broj ili obrnuto. Ovo deluje pomalo konfuzno, ali je

<table>
<tr><th colspan="2">Tabela 1</th></tr>
<tr><td colspan="2">Važniji operatori</td></tr>
<tr><td>+ - * / % ^</td><td>osnovne operacije</td></tr>
<tr><td>*= /= %= ^= += -=</td><td>operatori dodeljivanja sa operacijom</td></tr>
<tr><td>? c : e1 : e2</td><td>uslovni izraz</td></tr>
<tr><td>!</td><td>ne</td></tr>
<tr><td>||</td><td>ili</td></tr>
<tr><td>&&</td><td>i</td></tr>
<tr><td>== != = =</td><td>relacioni operatori</td></tr>
<tr><td>^</td><td>stepenovanje</td></tr>
<tr><td>++ --</td><td>povećavanje i smanjenje za 1</td></tr>
<tr><td colspan="2">Važnije funkcije:</td></tr>
<tr><td>atan, cos, sin, exp,
int, log, rand, sqrt</td><td>razne matematičke funkcije</td></tr>
<tr><td>gsub, sub, split</td><td>zamena i analiza teksta</td></tr>
<tr><td>index, match</td><td>traženje teksta</td></tr>
<tr><td>length</td><td>dužina stringa</td></tr>
<tr><td>sprintf</td><td>formatiranje stringa</td></tr>
<tr><td>substr</td><td>izdvajanje dela stringa</td></tr>
</table>

Važniji operatori i ugrađene funkcije jezika AWK

u radu i te kako korisno: nije potrebno ispitivati razne izuzetke i praviti posebne grane za obradu takvih slučajeva. Na primer, u jednoj datoteci imamo rezultate merenja, s tim što umesto broja može biti i oznaka "nema" i slično. Pošto se string "nema" konvertuje u broj 0, program za računanje prosečne vrednosti ima samo sledeće naredbe:

```
BEGIN { s = 0 , n = 0 }
{ n++ , s += $1 }
END { print s/n }
```

Tip promenljive određuje tip vrednosti koju ona trenutno sadrži, ali to i nije previše značajno kada se ima na umu da *AWK* veoma fleksibilno shvata pojmove "broj" i "string". Kada se u programu nađu operacije u kojima su operandi različitih tipova, onda se vrši konverzija:

```
x = 1
y = "2"
z = "3cde"

print x + y           # 3
print x - z           # 4
print x z             # "13cde"
```

Simbol # označava komentar: sav tekst naveden posle ovog simbola do kraja linije se ignoriše.

Nizovi u *AWK*-u su rešeni još duhovitije. Naime, indeks niza je string! To znači da je broj elemenata niza unapred nedefinisan i ograničen je jedino memorijom. Na primer, **a["xyz"]**, **a[$3]**, **a[234]** predstavljaju savršeno ispravno obraćanje elementima niza **a**! Naravno, sve operacije sa nizovima se obavljaju dinamički – drugačije ne bi ni moglo. Nema naredbi za kreiranje ili deklarisanje niza: jednostavno upotrebite neki element i kreiraćete niz.

Mogućnost da su indeksi niza stringovi ostavlja i neke probleme: kako opisati sve indekse nekog niza, tj. kako, na primer, odštampati sve elemente niza? Za to se koristi specijalna konstrukcija koja "menja" sve postojeće indekse. Može se koristiti na sledeći način:

```
if ("xyz" in a) print "Da"
for (i in a) print a[i]
```

Prva naredba ispituje da li se u nizu **a** nalazi element sa indeksom "**xyz**", a druga ispisuje sve elemente niza **a**.

Koliko je ova osobina dobra, pokazaće vam veoma ilustrativan program koji računa broj pojavljivanja svake reči u tekstu:

```
{ for (i = 1 , i <= NF , i++ ) freq[$i]++ }
END { for (i in freq) print i, freq[i] }
```

Razmislite koliko bi linija koda ovo zahtevalo na nekom klasičnom programskom jeziku.

Nizovi u *AWK*-u mogu biti samo jednodimenzionalni, ali vam više dimenzija uz ovakve indekse i nije potrebno: jednostavnom konkatenacijom indeksa možete simulirati nizove proizvoljnih dimenzija. Na primer, može se pisati a[1,2,3] – zarez "glumi" separator između indeksa.

Evo još jednog primera koji koristi nizove u *AWK*-u. Sledeći *AWK* program na osnovu rezultata komande **DIR** računa ukupan zbir datoteka po ekstenzijama:

```
(NF == 4 || NF == 5) && \
        $0 !  /<DIR>/ && \
        $NF   /:/ \
{ NF == 4 ? a["-"] += $2 : a[$2] += $3 }
END { for ( i in a) print i, a[i] }
```

U ovom primeru je korišćena mogućnost pisanja naredbi u više redova. Tada je potrebno na kraju reda koji se nastavlja staviti simbol "\". Slogovi koji sadrže podatke o datotekama imaju 4 ili 5 polja, u zavisnosti od toga da li datoteka ima ekstenziju. Da bi se izdvojili slogovi sa informacijama o datotekama, koristi se činjenica da ovi slogovi imaju simbol ":" u poslednjem polju (vreme kreiranja) i ne sadrže oznaku <DIR>. Na primer, za ovakav izveštaj:

```
 Volume in drive D is USER_DISK
 Volume Serial Number is 375A 18F2
 Directory of D:\TXT\RACUNARI\AWK

.            <DIR>       07-15-92  12:20p
..           <DIR>       07-15-92  12:20p
TEST                  12 08-12-92   8:28p
PRDBA    AWK         198 08-12-92   6:19p
INST                 134 08-12-92   5:08p
NAR      TXT        2122 08-12-92   4:05p
DOC      TXT         186 08-12-92   4:14p
A        AWK        2010 08-12-92   6:12p
AWK      WP        15432 08-12-92   8:29p
        9 file(s)      20083 bytes
                     5718016 bytes free
```

Rezultat rada programa je:

```
-        146
AWK     2209
TXT     2308
WP     15432
```

REZERVISANE PROMENLJIVE

Određene promenljive u *AWK*-u imaju posebno značenje i njihova vrednost se automatski menja. Neke od njih smo već koristili: **NR** je redni broj tekućeg sloga, **NF** broj polja u tekućem slogu. Evo ostalih interesantnijih sistemskih promenljivih:

ARGC	- Broj argumenata u komandnoj liniji
ARGV	- Niz koji sadrži argumente komandne linije
FILENAME	- Naziv datoteke koja se obrađuje
FS	- Separator polja u ulaznoj datoteci
RS	- Separator slogova u ulaznoj datoteci
OFS	- Separator polja u izlaznoj datoteci
ORS	- Separator slogova u izlaznoj datoteci

Veoma važna činjenica je da je **FS** regularni izraz koji u startu kao separator opisuje proizvoljan broj blanko simbola ili tabova. Međutim, to se može promeniti i tako potpuno izmeniti značenje pojma "polja". Na primer, ako obrađujete podatke koji su međusobno razdvojeni zarezima, na početku programa izvršite naredbu **FS = "[\t]*,[\t]*"** čime se označava da je ubuduće separator polja zarez oko koga je proizvoljan broj simbola blanko i tab. Ovo omogućava da za separator definišete proizvoljne simbole i tako veoma olakšate rad sa nekim grupama podataka.

Promenljiva **RS** sadrži separator koji razdvaja slogove (linije). Inicijalno je to "**\n**" (oznaka za novi red). Na žalost, ovde se ne može zadati proizvoljan regularan izraz, već samo "\n" i "" čime se slog povećava sa jedne linije na čitav pasus (do prve prazne linije). Ovo je naročito pogodno ako *AWK* koristite za obradu teksta.

Analogno se može menjati i izlazni format (promenljive **OFS** i **ORS**).

KO NE PROBA KAJAĆE SE

AWK je interpreter, što je potpuno razumljivo. Ipak, u praksi se pokazuje dovoljno brzim (naročito neke verzije). Usko grlo je uvek obraćanje disku, pa je brzina zadovoljavajuća, naročito kad se uzme u obzir da se *AWK* program obično koristi samo jednom: nešto vam zatreba, uradite to i zaboravite da ste ikada imali taj problem. Da li ćete problem rešiti za 2 ili 2.5 minuta, zaista nije previše značajno.

AWK program se napiše nekim standardnim editorom, a zatim se poziva interpreter komandom

AWK -f PROG.AWK DATA.TXT

Pri tome je **PROG.AWK** datoteka koja sadrži program, a **DATA.TXT** je datoteka koja se obrađuje. Rezultat se ispisuje na ekran, ali se redirekcijom lako usmerava u datoteku. Pošto su *AWK* programi obično veoma kratki, omogućeno je i zadavanje čitavog programa u komandnoj liniji:

AWK '{ print NR, $0 }' DATA.TXT

Da je *AWK* korisna stvar, to je van svake sumnje. Zaista je teško navesti sva oblasti obrade ASCII datoteka u kojima možete koristiti *AWK*. Može se samo reći da su to oni mali poslovi koje neprestano odlažete za sutra i obično ih nikada ne završavate: razne konverzije, zamene teksta, izmena formata itd. S obzirom na zaista minimalne zahteve (samo oko 50K prostora na disku), lako učenje i velike mogućnosti, zaista ćete pogrešiti ako ne isprobate ovaj programski jezik.

UMESTO KOPROCESORA

Porodica mikroprocesora 80x86 do skora nije podržavala rad sa racionalnim brojevima – ako je trebalo sabirati, deliti ili korenovati vrednosti koje nisu cele, ili izračunavati vrednosti elementarnih funkcija, morao se pisati odgovarajući, manje ili više komplikovan potprogram. „Intel" je, ipak, mislio i na one koji brz i efikasan rad sa racionalnim brojevima smatraju primarnom karakteristikom kompjuterskog sistema – njima se preporučuje da sistem dopune aritmetičkim koprocesorom 80x87. Počev od 80486, aritmetički koprocesor je integralni deo osnovnog procesora, ali se komunikacija sa njim i dalje poprilično razlikuje od „normalnog" toka stvari u okviru jednog procesora...

Dejan Ristanović

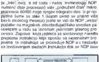
„Intel" je pre 80486 proizveo aritmetička koprocesore 8087, 80287 i 80387 – prvi je sarađivao sa 8088/8086, drugi (bazično) sa 80286, a treći prirodno sa 80386. Treba ipak znati da su 8087 i 80287 veoma slični, kako po konstrukciji tako i po karakteristikama – kao da „Intel" nije imao vremena da razvija potpuno novi čip. Neke od razlika su čak prevagnule na stranu pogoršanja performansi. Naime, komunikacija između 8086 i 8087 se obavlja posredstvom *Bus Interface Unit*-a ili BIU a po jedna ovakva jedinica ugrađena je kako u 8086 tako i u 8087. Kada 8087 primi naredbu, njegov BIU direktno pristupa memoriji da bi „pokupio" argumente i smestio rezultate. 8086 za to vreme može da izvršava sledeće instrukcije, pa čak i da pristupa memoriji. Integralni deo 80286 je, sa druge strane, takozvani MMU (*Memory Management Unit*) koji se, kao što smo pre mesec dana videli, bavi pristupom memoriji i omogućava adresiranje 16 megabajta realne i jednog gigabajta virtuelne memorije.

Da su „Intel"-ovi inženjeri želeli da komunikacija između 80286 i 80287 bude ista kao komunikacija između 8086 i 8087, morali bi da ugrade identičan MMU u 80287, što bi značilo gomilu novih tranzistora, veću disipaciju i samim tim neke teškoće sa kojima u datom trenutku očito nisu umeli da se uhvate u koštac. Zato je stvar maksimalno pojednostavljena: 80287 je dobio dva specijalna porta za komunikaciju preko jednog od DMA kanala procesora 80286, što znači da sam mikroprocesor šalje argumente i čeka rezultate, vrteći se u međuvremenu u nekoj vrsti mrtve petlje. Da bi bar delimično umanjio gubitke u brzini, „Intel" je omogućio priključenje asinhronog časovnika. Na žalost, IBM nije ugradio ništa slično u AT, a pa ni proizvođači klonova nisu hteli da se trude. Rezultat – 10 MHz 80287 na brzinskom *whetstone* testu postiže svega 320 000 što je čak i u poređenju sa 8087 veoma slabo. Da bi stvari bile još bizarnije, pokazalo se da aritmetički koprocesor koji, recimo, radi na 6 MHz zapravo **usporava** 80286 sa 10 na 9 MHz – mogu se napisati AT programi koji se brže izvršavaju **bez** koprocesora nego u njegovom prisustvu!

Dizajnirajući 32-bitni 80387 „Intel" je dobro pazio da ne ponovi greške – napravljen je zaokružen proizvod koji u potpunosti sarađuje sa nadređenim mikroprocesorom 80386. Vrline ovog koprocesora vide se i na brzinskom testu – na svega 16 MHz sistem postiže čak 1 800 000 *whetstones*. Zanimljivo je da je, verovatno zbog previsoke početne cene 80387 procesora, „Intel" obezbedio „saradnju" 32-bitnog 80386 sa 16-bitnim 80287, neke PC ploče naprosto omogućavaju da se utakne jedan ili drugi čip, premda ovakvo odricanje od performansi retko ima smisla.

INTEGRACIJA I OPTIMIZACIJA

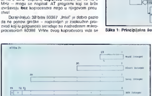
Integracija koprocesora u 80486 omogućila je dalja poboljšanja performansi – 25 MHz 80486 obezbeđuje oko 4 000 000 *whetstones*. Na raspolaganju je i eksterni koprocesor 80487SX koji „sarađuje" sa 80486SX – ovaj koprocesor je po mnogo čemu sličan 80387, pa su im i performanse približne. Treba ipak znati da je jedinica za rad sa racionalnim brojevima (u „Intel"-ovoj, a od sada i našoj terminologiji NDP, *numeric data processor*) više „pridruženi član" mikroprocesora 80486 nego njegov integralni deo. Iz principijelne šeme sa slike 1 (kojom ćemo se znatno više baviti sledećeg meseca) jasno se vidi da NDP „vreba" podatke na magistrali, obezbeđujući tako izvestan paralelizam u radu sa izvršnom jedinicom glavnog procesora. Zapravo, kada jedinica za izvršavanje naredbi prepozna instrukciju koja operiše sa racionalnim brojevima, ta instrukcija se prosleđuje NDP-u i nastavlja sa izvršavanjem sledećih instrukcija dok se NDP bavi aritmetikom. Nevolja nastaje ako je rezultat aritmetičke operacije potreban za dalji rad – tada nema druge nego se vrteti u „mrtvoj petlji" dok NDP ne dostavi rezultate. Kod 80486 ta sinhronizacija je potpuno automatska. Kod 80386 (i prethodnika) u kod se mora ugraditi instrukcija FWAIT (sinonim za WAIT) koju ćete svakako koristiti i kada radite sa 80486 ukoliko postoji i najmanja šansa da će se vaš program izvršavati na nekom 80386/387 sistemu.



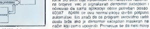
Slika 1: Principijalna šema 80486 sistema

Ukoliko programirate na nekom višem jeziku, instrukciju FWAIT po potrebi umeće vaš kompajler. Loš kompajler će je ugraditi posle svake aritmetičke instrukcije, obezvređujući tako čitav paralelizam koji NDP nudi. Prosečan kompajler umeće ovu instrukciju samo tamo gde je zaista potrebna. Optimizacioni kompajler će preurediti izraz ili čak niz naredbi tako da se paralelizam do maksimuma ispolji a instrukcija FWAIT postane retka. Naime, u prirodi je programera da zađeje nizove instrukcija koje se bave jednom vrednošću, a zatim „pređe" na sledeću, jer na taj način čovek razmišlja. Ti nizovi operacija se međutim u procesu optimizacije mogu na pogodan način „promešati", a da to ne utiče na krajnji rezultat.

U skladu sa „Intel"-ovom koncepcijom razvoja, svaki „sledeći" NDP treba da bude potpuno kompatibilan sa prethodnima, što garantuje izvršavanje jednom napisanog programa na svim budućim platformama. Potpuna kompatibilnost se, na žalost, ne može uvek postići, tako da između 8087, 80287, 80387 i 80486 postoje određene minorne razlike o kojima povremeno treba voditi računa. Pre svega, 8087 je signalizirao greške preko standardnog interapt mehanizma, dok 80387 obaveštava nadređeni procesor o grešci preko posebnog pina. 80386 na ove greške reaguje preko interapta 7, 9 i 16, dok 80486 koristi samo interapte 7 i 16. Instrukcije FENI i FDISI se dalje na 80387/80486 tretiraju kao NOP. No, najvažnija razlika je u tome što 8087 nije automatski normalizovao racionalne brojeve, već je signalizirao *denormal exception* i očekivao da sama aplikacija obavi potreban posao. 80387, 80486 će ovu normalizaciju izvršiti potpuno automatski, što znači da će program verovatno raditi dosta brže ako je *denormal exception* maskiran na način koji ćemo upoznati. Primećuje se da neki novi kompajleri ne generišu kod potreban za normalizaciju



Slika 2: Tipovi podataka koje NDP podržava



Slika 4: Racionalni broj u standardnoj tačnosti (*short real*)



Slika 5: Racionalni broj u dvostrukoj tačnosti (*long real*)



Slika 6: Racionalni broj u posebnoj tačnosti (*temp real*)

zbog čega prevedeni programi ne rade korektno na 8086/8087.

Razlike između 80287 i 80387/486 svode se na ukidanje instrukcije FSETPM (tretira se kao NOP) i nestanak takozvanog *projective closure* moda, o kome ćemo nešto kasnije govoriti. Osim pomenutog preusmeravanja interapta 9 na interapt 13, 80486 se razlikuje od 80387 po mehanizmu redirekcije grešaka preko NE flega u registru CR0, o kome smo govorili prošlog meseca. Hardverski, ova redirekcija se obavlja preko pinova FERR\ i IGNNE\.

FORMATI PODATAKA

Iako NDP bazično operiše sa 16-bitnim, 32-bitnim i 64-bitnim podacima, svi njegovi registri su 80-bitni. Zanimljivo je da se, kada se govori o koprocesoru, u „Intel"-ovoj dokumentaciji koriste različiti termini za cele brojeve – ono što se kod 80386 zvalo *dword* (32 bita) kod 80387 je *short integer*, dok je *quadword* (64 bita) postao *long integer*. Ostali tipovi podataka obuhvataju i BCD broj (80 bita), *short real* (32 bita), *long real* (64 bita) i *temp real* (80 bita). Upoznajmo detaljno svaki od ovih tipova podataka prikazanih na slici 2.

Najkraći podatak je 16-bitna reč (*word integer*). U prilogu je označen kao broj iz intervala [-32768, +32767] kodiran u potpunom komplementu. Za čitaoce koji se možda po prvi put sreću sa terminom potpuni (ili, kako se ponekad kaže, drugi) komplement, da kažemo da se radi o pogodnom načinu za predstavljanje negativnih brojeva. Uzmimo za primer broj 1101. To je, naravno, isto što i 13 ali je broj -1101 vrlo nepogodan za pamćenje u memoriji računara: osim jedinice i nule u njemu se pojavljuje i neka crtica koja predstavlja predznak. Ako pretpostavimo da svaki broj zauzima po jedan bajt memorije, mogli bismo da se dogovorimo da poslednji bit (prvi s leva) predstavlja kodirani znak tako da će se broj 13 pamtiti kao 00001101 a broj -13 kao 10001101. Ovakvo predstavljanje brojeva poznato je pod imenom „znak i apsolutna vrednost" i smatra se nepogodnim za mnoga primene, pa je izmišljeno nešto bolje – pamćenje komplementa broja. **Prvim** komplementom broja 13 (00001101) nazivamo binarni broj koji dobijamo kada umesto svake nule u tom broju stavimo jedinicu a umesto svake jedinice nulu: 00001101 tako postaje 11110010. Dodajući jedan na ovaj broj dobijamo 11110011, što je **drugi** ili potpuni komplement broja 13 i ujedno zgodan način da se predstavi broj -13.

Pokušajmo sada da izračunamo koliko je 18-13=18+(-13). Broj 18 binarno predstavljamo kao 00010010 a broj -13 kao 11110011. Sabiranjem ova dva broja dobijamo 100000101. Vidimo da rezultat ima 9 bita, premda smo radili samo sa 8; ako odbacimo deveti bit (pogledajte prošlomesečnu diskusiju o *carry* i *overflow* flegovima), rezultat postaje 00000101 ili, dekadno posmatrano, 5, što je, kao nekom čarolijom, isto što i 18-13. Iz ovog primera i svih primera koje ćete sami zamisliti može se zaključiti da je predstavljanje negativnih brojeva pomoću potpunog komplementa celishodno rešenje koje svodi oduzimanje na sabiranje sa negativnim brojem.

Ceo broj dvostruke tačnosti se naziva *short integer* i uopšte nije „kratak" – radi se o klasičnom 32-bitnom označenom celom broju koji većina kompajlera

Dekadno	Prvi komplement	Drugi komplement	Bias 7
-8	****	1000	0001
-7	1000	1001	0010
-6	1001	1010	0011
-5	1010	1011	0100
-4	1011	1100	0101
-3	1100	1101	0110
-2	1101	1110	0111
-1	1110	1111	1000
0	0000	0000	1001
1	0001	0001	1010
2	0010	0010	1011
3	0011	0011	1100
4	0100	0100	1101
5	0101	0101	1110
6	0110	0110	1111
7	0111	0111	****

Slika 3. Smisao *bias* koda

Znak	Eksponent	Mantisa	Vrednost
x	11...11	1 1xxx...xx	tihi NaN
x	11...11	1 0xxx...xx	signalni NaN
x	11...11	1 0000...00	beskonačno
x	00...00	0 xxxx...xx	denormal
x	00...00	0 0000...00	nula

Slika 7. Specijalne vrednosti u registrima NDP-a

naziva *long integer* (baš je „Intel" napravio terminološku zbrku!). Promenljiva ovog tipa može da primi ceo broj u intervalu [-2147483648, +2147483647].

Ceo broj četvorostruke tačnosti zove se *long integer* i obezbeđuje operacije sa označenim 64-bitnim celim brojevima iz intervala [-2^{63}, $+2^{63}-1$]. Ovakvi brojevi obično nastaju množenjem dva 32-bitna broja premda se mogu i samostalno koristiti za neka specifična numerička proračuna.

RACIONALNI BROJEVI

Glavna prednost „Intel"-ovog NDP-a primećuje se pri radu sa racionalnim veličinama – celi brojevi su, doduše, podržani iz više razloga, ali se sa njima sasvim dobro snalazi i „običan" 80386 ili 80486SX. Kada kažemo „racionalan broj", mislimo na ono što se u mnogim stranim i ponekoj domaćoj knjizi naziva **realnim** brojem. Odakle ova razlika? Racionalan je, pro sto rečeno, broj koji se može napisati kao količnik dva cela broja. Postoje i brojevi koji se (dokazano) ne mogu predstaviti na ovaj način, jedan od njih je koren iz 3, drugi je PI i tako dalje. Memorija računara može da „zapamti" **samo** brojeve koji imaju konačan broj decimala, a to su jedino *racionalni* brojevi; realne brojeva kao što je PI treba najpre aproksimirati racionalnima pri čemu se jasno gubi beskonačno mnogo decimala. Ako bismo hteli da budemo precizni do kraja, rekli bismo da se u memoriju računara može upisati samo element **konačnog** podskupa racionalnih brojeva (ovaj je skup jasno beskonačan mada ima manje elemenata nego skup realnih brojeva); obično se, međutim, zadovoljavamo terminom „racionalni brojevi". S obzirom da su svi racionalni brojevi ujedno i realni, neće mnogo pogrešiti onaj ko ove brojeve naziva realnima (ovaj termin verovatno potiče iz fortrana u kome se racionalni brojevi deklarišu sa *REAL*), treba samo znati o čemu se radi.

Veliki brojevi se često zapisuju pomoću eksponenta broja 10 – može se, na primer, reći da je brzina svetlosti 300 000 000 ili $3*10^8$ metara u sekundi. Broj 10 ovde potiče od dekadnog sistema u kome se izražavamo – 10^2 naprosto označava pomeranje decimalne tačke za dva mesta udesno a 10^{-2} za dva mesta ulevo ($5*10^{-2}$ je isto što i 0.05). S obzirom da računar radi sa binarnim brojevima, umesto osnove 10 treba koristiti osnovu 2, što znači da će se brojevi zapisivati u obliku $\pm f*2^e$ pri čemu smo slovom f označili mantisu a sa e eksponent. Broj $+0.11101*2^{101}$ se, na primer, sastoji od znaka plus, mantise 0.1101 („Intel" mantisu zove *significant* ali ćemo mi ostati pri uobičajenom terminu) i eksponenta 101.

Prezentacija bilo kog racionalnog broja u memoriji se takođe sastoji od znaka, eksponenta i mantise. Za znak je **uvek** rezervisan najviši (31-vi, 63-ći odnosno 79-ti) bit podatka, pri čemu nula na ovom položaju uvek označava pozitivan a jedinica negativan broj. Iz predstavljanja racionalnog broja u obliku „znak i apsolutna vrednost" sledi da se broj 0 može predstaviti na dva načina (+0 i -0) što je u nekim slučajevima zgodno a u drugima predstavlja smetnju.

Iza znaka sledi eksponent koji se kod *short*, *long* i *temp real* podataka kodira sa 8, 11 i 15 binarnih cifara respektivno. Negativni brojevi se predstavljaju u potpunom komplementu, ali uz jednu važnu razliku: od eksponenta se oduzima fiksan broj koji „Intel" naziva *bias* (kod nas se često koriste termini „kod sa nivoom" ili „kod sa viškom/manjkom"). Za racionalne brojeva u standardnoj tačnosti usvojen je *bias* 127, tako da se, na primer, eksponent 132 „pamti" kao 132-127 = 5. Za racionalne brojeve u dvostrukoj tačnosti usvojen je *bias* 1023 a za *temp real* 16383. Odgovor na pitanje zašto se od jedne logične predstave celog broja „nepotrebno" oduzima neka konstanta daje slika 3, na kojoj su predstavljeni brojevi iz intervala [-8, +7] najpre u potpunom komplementu, a zatim uz oduzimanje konstante 7. Vidi se da je ovom operacijom postignuto da brojevi, kada ih posmatramo kao neoznačene, budu sortirani istim redosledom kao i kada bi bili sortirani, tj. da -5 bude manje od -2. Na ovaj način se olakšavaju i ubrzavaju neke operacije sa racionalnim brojevima.

Eksponenti koji se sastoje samo od nula ili samo od jedinica (npr. vrednosti 00000000 i 11111111 kod *short real* promenljivih) se ne koriste za predstavljanje podataka, već označavaju neke specijalne stvari o kojima ćemo nešto kasnije govoriti.

Mantisa se smešta u memoriju kao „običan" binarni broj, ali tu postoji jedna specifičnost. Kada smo napisali brzinu svetlosti u obliku $3*10^8$, iskoristili smo njenu uobičajenu predstavu, ali smo isto tako mogli napisati i $30*10^7$, $300*10^6$ ili čak $0.3*10^9$. Konvencijom je predviđeno da se racionalni brojevi **uvek** predstavljaju u obliku $s_0.s_1s_2s_3s_4s_5...s_n$, pri čemu je $s_0=1$. To znači da binarni broj najpre treba šiftovati potreban broj puta uz odgovarajuće povećanje ili smanjenje eksponenta, dok se on ne dovede na željeni oblik, a tek tada smestiti u memoriju. Autori IEEE 754 formata koji „Intel"-ov NDP koristi su se, međutim, „dosetili" da uštede jedan bit – pošto je s_0 po definiciji uvek 1, nema nikakvog smisla pamtiti ga, tako da se kod *short* i *long real* podataka memorišu samo $s_1s_2s_3s_4s_5...s_n$ dok se vodeća jedinica i decimalna tačka podrazumevaju. Kod *temp real* podataka nije tako – memoriše



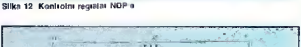
Slika 8. Binarno kodirani decimalni brojevi (BCD)



Slika 9. Registri NDP-a



Slika 11. Statusni registar NDP-a



Slika 12. Kontrolni registar NDP-a



Slika 13. Registri ukazatelji greške

se i vodeća jedinica dok se tačka prirodno podrazumeva iza te prve cifre. *Temp real* tip se u stvari koristi pri računanju: u njega se „raspakuje" standardni racionalan broj a onda račun ide mnogo brže. Pošto je operacija završena, broj se ponovo „normalizuje" (dovodi u oblik $1.s_1s_2s_3s_4s_5...s_n$) i smešta u memoriju bez vodeće binarne jedinice.

Da rezimiramo: racionalan broj u standardnoj tačnosti (slika 4) obezbeđuje operacije sa brojevima čija je apsolutna vrednost u intervalu $(1.18*10^{-38}, 3.4*10^{38})$ uz rad sa oko sedam tačnih cifara, dvostruka tačnost (slika 5) podrazumeva interval apsolutnih vrednosti $(2.23*10^{-308}, 1.79*10^{308})$ uz 15 tačnih cifara, a *temp real* (slika 6) interval $(3.37*10^{-4932}, 1.18*10^{4932})$ uz oko 19 tačnih cifara. U svim ovim slučajevima se osim brojeva iz pomenutih intervala može memorisati i poneka specijalna vrednost.

BROJEVI KOJI TO NISU

Jedna od tih specijalnih vrednosti je „obična" nula koja se predstavlja kao proizvoljan znak (0 ili 1 umesto plus ili minus) i potreban broj binarnih nula – *short real* nula, na primer, može da se sastoji od 32 binarne nule ili jedne binarne jedinice i 31 binarne nule. Druge važne specijalne vrednosti su „brojevi" čiji se eksponenti sastoje od samih nula ili samih jedinica. Eksponent od jedinica i mantisa 1.0000...0 označava „beskonačno" koje, zavisno od znaka, može biti „plus beskonačno" ili „minus beskonačno". Vodeća jedinica se, kao što smo videli, ne piše kod racionalnih brojeva u standardnoj i dvostrukoj tačnosti, tako da broj plus beskonačno upisan u polje standardne tačnosti glasi

01111111100000000000000000000000

Ovakva nomenklatura je sasvim u skladu sa IEEE standardima koji zahtevaju da, primera radi, 1/0 signalizira grešku, ali da rezultat u odgovarajućem registru bude smislen i da ta vrednost bude „beskonačno".

Ukoliko se eksponent sastoji od jedinica a mantisa je različita od 1.00...00, radi se o ne-broju NaN (skraćenica od *not a number*). 80387 poznaje „tihe" (*quiet*) i „alarmne" (*signaling*) ne-brojeve koji se razlikuju na osnovu najsignifikantnijeg bita razlomljenog dela mantise (slika 7). Ukoliko se u bilo kojoj aritmetičkoj operaciji pojavi alarmni ne-broj, koprocesor će generisati odgovarajući interapt. NDP će automatski generisati tihi ne-broj ukoliko je tražena operacija nemoguća.

Za šta služe alarmni ne-brojevi? Većina savremenih programskih jezika ne dopušta programeru da koristi promenljive kojima prethodno nije dodeljena vrednost. Na prvi pogled bi se reklo da ovakav zahtev predstavlja nepotrebno ograničenje (na mnogim verzijama bejzika referenciranje nedefinisane promenljive povlači njeno automatsko definisanje i dodeljivanje neke karakteristične vrednosti, obično nule) ali se u stvari radi o velikoj pogodnosti koja olakšava pronalaženje grešaka u programu – pogrešno uneseno ime promenljive ili neka slična zamena dovodi do problema koje je vrlo teško locirati! Kako se tehnički realizuje registrovanje nedefinisanih promenljivih? Kada se radi o interpretatoru, problema nema: u trenutku izvršavanje naredbe proverava se tekuća vrednost promenljivih i uvek se zna šta je definisano a šta nije. Kod kompajlera je, međutim, problem gotovo nerešiv pošto se obično još u toku prevođenja alocira statički prostor za promenljive. Kompajler će jasno prijaviti promenljive koje su referencirane iako nikada nisu definisane, ali nema nikakvog načina da sazna da li će u toku izvršavanja programa neka promenljiva biti najpre pozvana a onda definisana! U pomoć priskaču alarmni ne-brojevi – kompajler svakoj promenljivoj u startu dodeljuje istu ne-vrednost, a onda se NDP „buni" čim program pokuša da izvrši bilo koju operaciju čiji argument nije ispravan. Rezultat je sigurnije programiranje i lakše pronalaženje grešaka, bez gubitka na performansama koji bi nastao ako bi kompajler u svaki aritmetički izraz ugrađivao kod koji poredi sve korišćene promenljive sa nekom softverski definisanim ne-brojem.

Ostalo je još da pomenemo brojeve koji su suviše mali da bi se uspešno normalizovali i upisali u memoriju. U „Intel"-ovoj terminologiji zovu se *denormals* i predstavljeni su nultim eksponentom i mantisom različitom od nule (slika 7).

```
x      DT     ?           ; x u standardnoj tačnosti
y      DD     ?           ; rezultat
const  DW     ?           ; prostor za celobrojne konstante
       FLD    x           ; x na stek
       FLD    ST(0)       ; duplira x
       FMUL   ST(1)       ; x^2 na vrhu steka
       MOV    const, 3    ; celobrojni množilac
       FIMUL  const       ; x^2*3 na vrhu steka
       MOV    const, 7    ; celobrojni množilac
       FILD   const       ; 7 na vrhu steka
       FMULP  ST(2), ST   ; ST(2)=7x, pop steka
       FSUBRP ST(1), ST   ; ST(1)=3x^2-ST(1), pop steka
       MOV    const, 4    ; celobrojna konstanta
       FIADD  const       ; 3x^2-7x+4 na vrhu steka
       FSTP   y           ; rezultat sa steka u memoriju
```

Slika 10. Kako NDP radi sa stekom

BINARNO KODIRANI DECIMALNI BROJEVI

NDP obezbeđuje i rad sa binarno kodiranim celim brojevima (BCD) koji se upisuju u 10 susednih bajtova memorije u skladu sa slikom 8. Najviši bit je i dalje znak (1 označava negativan broj), sledećih sedam bita se ne koriste, a ostatak su decimalne cifre kodirane klasičnim 8421 kodom sa po dve cifre u svakih 8 bita. Memorišu se, dakle, brojevi iz intervala

(-999999999999999999, +999999999999999999)

Mnogi početnici smatraju binarno kodirane brojeve za nekakvu „prirodnu" računarsku reprezentaciju podataka na koju smo navikli. U većini slučajeva, na žalost, ovakvo kodiranje nije pogodno i to ne toliko zbog redundance (za svaku cifru koristimo po četiri bita, što znači da umesto 16 cifara koje bismo mogli da „pamtimo" koristimo samo 10) koliko zbog mukotrpnog i sporog računanja koje mora da „imitira" ljudski rad na način krajnje nepogodan za kompjuter – ako ne verujete, pokušajte da nađete logaritam nekog broja računajući „na ruke". Pa ipak, binarno kodiranje decimalnih brojeva ima dobrih strana od kojih je jedna bila dovoljna da ga preporuči za neke jezike kao što je COBOL. Osnovna dobra strana je tačnost: ukoliko petljamo sa novcem, neobično nam je važno da se računi slažu do poslednje pare. Danas preovlađujući rad sa brojevima u pokretnom zarezu, kao što smo videli, unosi određenu grešku usled zaokruživanja, dok je takva greška kod rada sa BCD brojevima strogo kontrolisana: ako radimo sa dva decimale, uvek ćemo imati dve tačne decimale! Druga dobra strana BCD kodiranja se primećuje kod prenosa podataka: znajući da se za „pamćenje" brojeva koriste jedino binarne kombinacije 0000 – 1001, otkrivanje bilo koje neregularne kombinacije pokazuje da je došlo do greške u komunikaciji i da je primljeni broj besmislen.

REGISTRI NDP-a

Arhitektura NDP-a obuhvata osam 80-bitnih registara (slika 9) koji se ne adresiraju po imenu nego čine stek. Prvi registar steka (odnosno poslednja vrednost poslata na stek) se obično naziva ST ili ST(0), sledeći registar (prethodno poslata vrednost) je ST(1) i tako dalje, sve do ST(7). Sukcesivnim operacijama operandi se smeštaju na stek a onda zamenjuju rezultatima. Sve je to lako razumeti uz pomoć slike 10. Na početku rada je ST(2) definisan kao vrh



steka, ali se na njemu ne nalaze nikakvi podaci. 16-bitni registar TW (*tag word*) „čuva" po dva bita za svaki od registara ST(0), ST(1), ... ST(7) na osnovu kojih se zna da li je u dati registar upisan neki podatak, nula, specijalna vrednost ili je sadržaj nedefinisan. Na slici 10 je prikazan program koji računa vrednost polinoma $y = 3x^2 - 7x + 4$.

Struktura statusnog registra (**status word register**) prikazana je na slici 11. Ovde ćemo ukratko upoznati značenje svakog od njegovih bitova.

B (busy) je 1 dok NDP izvršava neku instrukciju. Ovaj bit možete da ispitate komandom FNSTSW AX koja kopira statusni registar u AX, gde se nepotrebni bitovi mogu lako maskirati.

C3, C2, C1, C0 (condition codes) opisuju operaciju koju NDP trenutno izvršava: poređenje, testiranje, aritmetička instrukcija... Ovde nećemo trošiti prostor na konkretne vrednosti koje opisuju pojedina stanja pošto su one u praksi vrlo retko potrebne – čitaoce koje zanimaju detalji upućujemo na „Intel"-ovu dokumentaciju.

TOP (top of stack) ukazuje na registar koji trenutno predstavlja vrh steka. Kada na stek dođe nova vrednost, TOP se smanjuje za 1, dok skidanje vrednosti sa steka uvećava TOP za jedan. Posle svake od ovih operacija TOP se „odseca" na tri bita – mehanizam koji signalizira prekoračenja steka zasniva se na flegu SF.

ES (error summary) se postavlja na 1 kada god neka instrukcija izazove nemaskirani prekid (*exception*). Opis nastale greške se može pročitati iz bitova 0-5 ovog registra.

SF (stack fault) se postavlja na 1 ako korisnik stavi više od 8 podataka na stek ili pokuša da sa steka skine podatak koji prethodno nije tamo „deponovan". Ovaj fleg ne postoji na 80287, tako da na njega ne treba da računate ako pišete program koji će se izvršavati na svim 80x86 platformama.

Pre nego što razmotrimo svaki od preostalih flegova, da objasnimo princip njihovog delovanja. Svaki od ovih flegova označava jednu od grešaka koje mogu da nastupe u toku izvršavanja neke aritmetičke operacije. Ako takva greška nastupi, NDP upisuje 1 u odgovarajuće polje ovog (statusnog) registra, a zatim proverava odgovarajuće polje kontrolnog registra. Ako tamo nađe nulu, greška je kritična i izvršava se interapt koji će je obraditi. Ako je odgovarajući bit kontrolnog registra 1, greška se zanemaruje i prelazi na izvršavanje sledeće instrukcije. Međutim, jednom setovan bit u statusnom registru ostaje „večito" setovan (dok program eksplicitno ne upiše novu vrednost u njega) što obezbeđuje da se na kraju izvršavanja programa ili nekog njegovog segmenta proveri kakvih je sve maskiranih (zanemarenih) grešaka bilo.

PE (precision exception) nastupa gotovo neprekidno, tj. kada god neka aritmetička instrukcija izazove grešku u zaokruživanju. Čak i sasvim obična računanja poput 1/3 izazivaju ovu grešku, baš kao i konverzije podatka iz *temp* u *long real* uz gubitak neke tačne cifre. S obzirom da se gotovo i ne može zamisliti program koji radi sa racionalnim brojevima i u koma se ovakva zaokruživanja ne događaju, ovaj će interapt uvek biti onemogućen a PE u status registru će imati vrednost 1. Interapt se ponekad omogućava pri radu sa BCD brojevima, ako primena ne dopušta baš nikakav gubitak tačnosti.

UE (underflow exception) nastupa kada je rezultat neke operacije suviše mali da bi bio registrovan. Na primer, najmanja apsolutna vrednost (različita od nule, naravno) koja se može upisati u *temp real* promenljivu je oko $3.37*10^{-4932}$. Ako, recimo, izvedete koren iz ovog broja, dobićete *underflow exception*.

OE (overflow exception) nastupa kada je rezultat neke operacije prevelik da bi bio registrovan. Kao i PE, UE i OE mogu da nastupe i pri običnoj konverziji, na primer pri prepisivanju *temp real* podatka u *short real* polje.

ZE (zero divide exception) nastupa ako od NDP-a zatražite da podeli neki broj nulom. Može nastupiti i kod nekih nedefinisanih tangensa i sličnih operacija.

DE (denormal exception) nastupa kada se tokom računanja javi broj koji se ne može normalizovati. O ovakvim brojevima (bolje reći ne-brojevima) smo već govorili.

IE (invalid operation exception) obuhvata sve greške koje nisu „pokrivene" nekim od prethodnih flego

NASTAVLJA SE NA STRANI 69

SLOVO NA SLOVO

Inicijativu za ovaj tekst dao je urednik „Računara", smatrajući da je područje fontova nedovoljno zastupljeno u časopisu. Iako ne mislim da sam autoritet u ovoj oblasti, dugogodišnje iskustvo u radu sa fontovima, brojni problemi na koje sam nailazio i koje sam rešavao, kao i ne mali obim pročitane literature, daju mi za pravo da smatram aebe dovoljno stručnim da pišem o ovoj temi.

Milan Velimirović

Upozoravam da mi je struka programiranje, a ne tipografija, i da na ovu oblast verovatno gledam kroz drugačiju prizmu od autora prošlogodišnje serije članaka u „Računarima" (Robert Slovački: *"Iz mita u legendu"* 73/91, *"Rat fontova"* 74/91, *"Type 1 ili True Type"* 75/91).

Istina, želja mi je da fontove kreiram više sa estetske nego sa tehničke strane, više očima grafičara nego informatičara. Zato ovde neće biti reči o „večitom" (po mom mišljenju i pomalo „naduvanom") problemu kodnog rasporeda. Prateći relativno kratak istorijat, pokušaću da ukažem na probleme koje je potrebno prevazići da bi se dobio kvalitetan font, kao i na rešenja kojima su različiti proizvođači pribegavali. Cilj ovog članka je da pomogne kako iskusnim dizajnerima slova koji još uvek nemaju dovoljno iskustva u digitalnoj tipografiji, tako i iskusnim „kompjuterašima" koji imaju malo grafičkog iskustva.

MEMORISANJE OPISA FONTA

Font se može zapisati na dva osnovna načina: u rasterskom (bit-mapiranom) i vektorskom (konturnom) obliku. Postoje još neki načini opisa slova, ali oni uglavnom predstavljaju varijaciju ili svojevrsnu kombinaciju bazičnih oblika.

Bit-mapa. Rasterski ili bit-mapirani fontovi predstavljaju definiciju svake pojedinačne tačke u pravougaoniku koji uokviruje određeno slovo. Binarna nula obično označava belu, a binarna jedinica crnu tačku. O ovom formatu, konkretno o HP SFP fontovima, bilo je reči u ranijim brojevima „Računara", pa ne treba ovde gubiti vreme na ponavljanje već rečenog.

Bajt-mapa. Ovaj oblik u suštini je analogan bit-mapi, dok se razlika primena ogleda u eventualnom definisanju polutonova ili boja određenih tačaka.

Linijska mapa. Slobodan prevod engleskog naziva *Run length format*. I ovo se svodi na bit-mapu, s tim što se definiše početna tačka i broj uzastopnih belih, odnosno crnih tačaka za svaku (horizontalnu ili vertikalnu) liniju rastera, tj. sken-liniju. Pretpostavlja se da linija počinje izvan slova, tako da je prvi segment svake sken-linije beo, zatim sledi promena u crno, pa u belo itd. Proračuni su pokazali da je za kodiranje slova za manje rezolucije i manje punktaže ekonomičniji bit-mapirani zapis, a sa porastom rezolucije ili veličine slova odnos se menja u korist linijske mape.

Otvoreni i zatvoreni vektori. Format je retko korišćen u praksi, a predstavlja kombinaciju bit-mapiranog i vektorskog formata. Suština **otvorenih vektora** je u tome da se tačke prelaska belog u crno i obratno definišu kombinacijom početnih i krajnjih tačaka niza vektora u kombinaciji sa linijskim mapama na pozicijama gde postoji prekid u lancu vektora (vektor koji ne počinje u nastavku prethodnog ili sa na njega ne nastavlja novi naziva se **otvoreni vektor**). Ovakav način opisa nije praktičan, recimo, za rotaciju, zbog fiksnih horizontalnih ili vertikalnih linijskih mapa. U slučaju da se izostave linijske mape i celo slovo definiše konturama sačinjenim od lanaca vektora, reč je o **zatvorenim vektorima**, odnosno o nizu linijskih segmenata.

Konturni fontovi. Svi fontovi kod kojih se memorišu samo granične tačke između crnih i belih površina, odnosno kontura slova, mogu se svrstati u ovu grupu. Ipak, u praksi se ovaj termin odnosi na one formate kod kojih su definisane karakteristične tačke na konturi i matematičke funkcije pomoću kojih se izračunavaju nizovi tačaka koji ih spajaju. Kod već spomenutih zatvorenih vektora to su isključivo linijski segmenti (vektori): pamte se samo koordinate u kojima kontura menja smer, a konture se izvlače jednostavnom interpolacijom između susednih tačaka – izuzetno brz proces. Nevolja je, međutim, što bi kontura sa dosta krivina zahtevala veliki broj veoma bliskih koordinata i veoma opširna definicije. Takvi delovi mogu se mnogo ekonomičnije opisati kao složenije funkcije, recimo kao delovi kružnice (lukovi) kod kojih je potrebno zadati samo početnu i krajnju tačku i/ili centar kružnice.

Termini „konturni" i „vektorski" fontovi u praksi se poistovećuju.

ŠTA JE BOLJE?

Bez obzira kako je font zapisan, slovo se na papir prenosi u vidu gusto grupisanih tačaka, znači u rasterskoj formi. I dok je kod bit-mapiranog fonta rasterizacija izvršena već prilikom njegovog kreiranja, kod konturnog fonta rasterizacija se vrši automatski u procesu štampe. Poređenje dva načina zapisa dovodi do zaključka da i jedan i drugi imaju svoje prednosti, ali i svoje nedostatke. Uzmimo nekoliko kriterijuma za poređenje.

Prostor. Nemogućnost uspešnog skaliranja ili deformisanja rasterski definisanog fonta zahteva da za svaku veličinu i svaki oblik pisma postoji definicija kompletnog fonta. Tako na primer, za Hewlett Packard bit-mapirani (SFP) font u 10 najčešće potrebnih veličina (6, 8, 9, 10, 11, 12, 14, 16, 18, 24, 30 punkta) u četiri oblika (normal, bold, italic i bold-italic) treba računati na zauzeće od oko 1MB eksterne memorije. Kod konturnih fontova memoriše se samo „master" zapis, koji u procesu štampe može biti skaliran na bilo koju veličinu bez gubitka kvaliteta. Za font koji odgovara pomenutom HP SFP fontu potrebno je čuvati samo mastere za četiri oblika, što bi, recimo za Adobe Type 1 (PostScript) format, zauzelo oko 200K prostora, uz napomenu da je i raspon punktaža dramatično povećan – od jedne pa do više od 1000 tačaka, sa mogućnošću postizanja i međuveličina (recimo 9.5 tačaka). U pogledu zauzeća prostora prednost je očigledno na strani konturnog zapisa.

Kvalitet. S obzirom da se prenos na papir (film) vrši po sistemu „tačka za tačku", kvalitet štampe rasterski zapisanog fonta identičan je kvalitetu samog zapisa. Kod vektorskih fontova to nije slučaj, naročito pri malim punktažama. Ovde kvalitet štampanog slova zavisi i od kvaliteta algoritma koji vrši rasterizaciju, ali i od rezolucije izlaznog uređaja. Treba imati u vidu da za generisanje idealnog slova nije dovoljno imati samo precizan matematički proračun – često je potrebno, u cilju postizanja utiska idealnog, odstupiti od idealnog. Dizajn slova je prava nauka koja, između ostalog, ne zanemaruje ni fiziološki element, manu oka, fenomen optičke varke. Ovo ne spominjem sa namerom da ulazim u objašnjenje fenomena, već samo da bih nagovestio koliko inteligencije treba očekivati od algoritma čiji je posao na prvi pogled jednostavan: skaliranje (odnosno množenje i deljenje), izvlačenje i popunjavanje kontura. Algoritam mora da bude sposoban da izvrši neophodne korekcije, o čemu će biti reči kasnije u ovom tekstu. U ovom trenutku treba reći da je sa smanjenjem punktaže problem sve izraženiji, jer ostaje mnogo manje raspoloživih tačaka za korekcije, a da povećanje rezolucije, odnosno broja tačaka na istoj površini, olakšava posao. Ipak, može se zaključiti da bit-mapirani fontovi u principu treba da garantuju bolji kvalitet, s obzirom da ovaj slučaj spada u sve manji broj izuzetaka kada je ljudski mozak nadmoćan u odnosu na mašinu. Naročito i ne bez razloga proizvođači fontova deklarišu da je njihov proizvod ručno dorađen.

Brzina rada. Po ovom kriterijumu rasterski fontovi su do nedavno bili izrazito nadmoćni. No, svedoci smo neverovatno brzog razvoja tehnologije koji je gotovo doveo do izjednačavanja brzine štampe rasterskih i vektorskih fontova.

Fleksibilnost. Ovde je prednost konturnih nad rasterskim fontovima najveća. Ne postoji dobar način da se slika u bit-mapi poveća, smanji, deformiše ili (pogotovu) rotira bez vidljivog gubitka preciznosti. Sa konturnim fontovima to je „mačji kašalj" – za najveći broj promena dovoljne su operacije množenja i deljenja, a i komplikovaniji zahtevi kao što su tekst po kružnoj baznoj liniji, unutar konture, pa čak i po deformisanoj površini, nisu nerešiv problem.

Wysiwyg. Uzimanje ovog elementa za jedan od kriterijuma pri poređenju zapisa možda izgleda malo isforsirano. Međutim, usaglašavanje izgleda na ekranu i rezultata na papiru i te kako je bitan uslov kako za povećanje brzine rada, tako i za smanjenje utroška materijala. Ovde su konturni fontovi očigledno neuporedivo praktičniji – naravno, pod uslovom da ih aplikacija koristi za kreiranje ekranskog prikaza. Korisnici GEM verzije *Venture* sa HP Laser Jet printerom (ubedljivo najčešća kombinacija!) neka se prisete mukotrpnog rada na kreiranju ekranskih fontova koji koliko-toliko odgovaraju printerskim, notorne potrebe za njihovo prilagođavanje pri svakom nestandardnom tekstu itd. Uporedi li se to, na primer, sa *Venturom* za *Windows*, pa možda i sa instaliranim Adobe Type Managerom, videće se koliko je olakšanje i pojednostavljenje.

KRIVI SEGMENTI

Pre no što pokušamo da odgovorimo na pitanje „Kako dobiti kvalitetan konturni font?" moramo prihvatiti činjenicu da dobar font može da napravi samo ljudska ruka. Nema toliko dobrog algoritma koji bi mogao da nadmaši ljudski mozak, ljudsko oko ili ruku. Zato prvi deo odgovora leži u praktičnosti formata fonta koji mora da omogući dve osnovne stvari: 1) fleksibilnost pri intervencijama u toku procesa dizajniranja i 2) brzu i preciznu obradu u procesu štampe. Kritičan deo su funkcije kojima se definišu krive linije. Koje?

Luk. Formati koji koriste lučni segment najčešće ga opisuju pomoću krajnjih tačaka i centra kružnice čiji je luk deo, dok se kod nekih jedan ili obe krajnje tačke zamenjuju definicijom ugla, odnosno tangensa. Luk kao element veoma je jednostavan za imple-

mentaciju kako u procesu dizajniranja tako i u procesu štampe, ali samo ako se posmatra nezavisno od susednih segmenata konture. Problemi nastaju kod spojnih tačaka, gde dizajner mora da uloži mnogo napora kako bi postigao kontinuitet konture. Poznati formati koji krive definišu lučnim segmentima su *Bitstream Fontware* (BCO), *IntelliFont*, *URW* (DI, VC formati).

Spirala, odnosno deo spirale, je jedan od načina prevazilaženja problema kontinualnosti. Vic je u tome da se uzorci krivih (poznati kao *francuske krive*) koji su delovi spirale, poklapaju sa putanjom konture. Čini mi se da ovaj format bar u oblasti DTP nije primenjivan, a konstruišu ga nama manje poznati proizvođači (*Purdy*, *Ilak*).

Bezijerove krive i splajnovi. Splajnovi (*Splines*) su najrašireniji i najfleksibilniji način definisanja krivih linija. Bezijerove krive koriste formati kao što su *Adobe Type I*, *Bitstream Speedo* (BEZ), *URW Nimbus Q*, a žalim i PC korisnicima najbliži *Corel DRAW!* (WFN) ili *ZSoft Publisher's Type Foundry* (OTL) format. Splajnovi se sreću kod nekih *XEROX* formata, a i kod novog Microsoft/Apple *TrueType* standarda. Neću se upuštati u njihovo objašnjavanje, već čitaoca upućujem na odličan tekst „Bezierova krivica" (Računari 76/1992, strana 46). Istina, algoritam za interpolaciju krivog segmenta je složeniji od onog za lučni segment, ali ne toliko da bi potro prednosti dobijene krajnjim pojednostavljenjem posla dizajnera. Pošto kubni polinom koji je naveden u spomenutom tekstu deluje zastrašujuće za one koji su zaboravili (ili nikad nisu znali) matematiku, ovde ću opisati alternativni, lakše shvatljiv i za implementaciju neuporedivo jednostavniji način interpolacije. To je metod subdivizije Bezijerove krive.

Na slici 1 prikazana je Bezijerova kriva sa početnom tačkom $A_{(x1,y1)}$ i krajnjom tačkom $D_{(x3,y3)}$, dok je njen tok kontrolisan tačkama $B_{(x1,y1)}$ i $C_{(x2,y2)}$. Poligon (A, B, C, D) predstavlja kontrolni poligon Bezijerove krive. Metoda subdivizije sastoji se u tome da se najpre izračuna srednja tačka (koja u polinomu odgovara parametru u=0.5). Najpre se izračunaju srednje tačke tri strane kontrolnog poligona:

$$B_1 = \frac{AB}{2} \qquad C_1 = \frac{BC}{2} \qquad D_1 = \frac{CD}{2}$$

Zatim se podele linije između dobijenih tačaka

$$B_2 = \frac{B1C1}{2} \text{ i } C_2 = \frac{C1D1}{2}$$

I konačno podeli linije između na taj način dobijenih tačaka

$$B_3 = \frac{B2C2}{2}$$

Tačka B_3 pripada željenoj krivoj. Kod programa za opisanu operaciju krajnje je jednostavan: sastoji se od 6 sabiranja i 6 šiftovanja udesno (deoba sa 2). Dalja subdivizija može se vršiti metodom u programiranju poznatom kao rekurzija. Subdivizijom krivih (A, B, B_2, B_3) i (B_3, C_2, D, D) dobijaju se novi parovi za subdiviziju, i tako dalje. Pogodnost ove metode je u tome što se rezultujuća kriva može aproksimirati kao niz kontrolnih poligona, između kojih se vrši interpolacija linijskih segmenata. Na primer, u našem primeru već nakon prve subdivizije poligon (A, B1, B2, C2, D1, D) dovoljno verno prati tok krive. Svaki dalji nivo subdivizije rezultira još vernijim poligonom, pa se (recimo, za potrebe bržeg prikaza na ekranu prilikom dizajniranja) može prekinuti i mnogo pre dostizanja najlinijeg oblika, čime se znatno povećava brzina interaktivnog rada.

OD DIZAJNA SLOVA DO ŠTAMPE...

Font lajl proizvodi se na osnovu novog dizajna nacrtanog na papiru a možda i na računaru, ili se može iskonstruisati neko već postojeće pismo. U svakom slučaju, potrebno je najpre digitalizovati master, a to je najteži deo posla koji se još uvek mora uraditi „peške". Naime, nijedan ozbiljan dizajner slova ne pomišlja na razne *auto trace* alate. Svako slovo crta čovek, sve ostale elemente opisa fonta takođe određuje čovek. *Auto trace* je omiljena dokolica inženjera u ovoj oblasti, a rezultati jedva da mogu da zadovolje kvalitet *shareware* fontova.

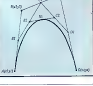
Koordinatni sistem (em-kvadrat). Opis slova sastoji se od jedne ili više kontura sastavljenih od pravih i krivih linija, a pamte se koordinate karakterističnih tačaka. Najmanja adresabilna jedinica u koordinatnom sistemu naziva se na engleskom FUnit (ovde ćemo koristiti skraćenicu FU). Sistem ima koordinatni početak (0,0), ali iz razumljivih razloga nije neograničena ravan. Ako pretpostavimo da se koordinate memorišu kao 16-bitne rači, e da mogu da imaju i negativni predznak, jasno je da koordinata ne može imati vrednost veću od +32735 FU, a manju od -32636 FU.
To još uvek višestruko prevazilazi praktične potrebe, za koje je sasvim dovoljan raspon -1000...+1000 FU.

Koordinatni početak nema suštinsku ulogu za opise slova. Time se želi reći da koordinata x=0 nema neko posebno značenje, dok se za koordinatu y=0 obično pretpostavlja da odgovara baznoj liniji fonta, ali ni to nije obavezno. Na primer, opis se može postaviti tako da x vrednost 0 predstavlja estetski centar slova. Ovo je slučaj kod katakana i drugih pisama koja se slažu vertikalno.

Druga jedinica mere na koju treba skrenuti pažnju je tzv. *em*, odnosno Em kvadrat. Ni ova jedinica nema fiksnu vrednost, već se određuje prema karakteristikama fonta, a varira pri skaliranju fonta. Najčešće je to kvadrat sa bazom jednakom širini najšireg slova u fontu, a to je obično veliko slovo „M" - otud i naziv „em". I dok u doba olovnog sloga slova nisu mogla da se prostiru izvan em kvadrata, kod digitalne tipografije nema tog ograničenja. Em kvadrat može da bude izabran tako da bude dovoljno velik da primi svako slovo, uključujući i slova čiji su delovi ispod bazne linije (j, p, y...) ili akcentirana slova (poput naših Ć, Č, Š, Ž). Ili, ako se to pokaže prikladnim, može da bude manji i da se delovi pojedinih slova prostiru izvan kvadrata. Većina formata dozvoljava oba pristupa, tako da je odluka stvar proizvođača fonta. Tipična em vrednost je nešto veća od pune visine, da bi se izbeglo sudaranje slova u susednim linijama teksta pri normalnom proredu.

Granularnost je određena brojem *FU* po *em* kvadratu (*UPM*). *FU* je relativna vrednost, pošto *UPM* ostaje isti kada se *em* menja prilikom skaliranja. Određena tačka u slovu imaće istu koordinatnu poziciju izraženu u *FU*, bez obzira na punktažu za koju se font priprema u procesu štampe. Ovo je pogodno zbog toga što se sva podešavanja mogu izvršiti na originalnom opisu, a jednoobrazno primeniti na bilo kojoj veličini pisma.

Skaliranje slova. Bez obzira na granularnost



em-kvadrata korišćenog za definisanje fonta, pre nego što se slovo odštampa (ili prikaže na ekranu) ono mora da bude skalirano tako da odgovara karakteristikama izlazne jedinice. Skaliran opis mora da opisuje konture u absolutnom sistemu mere koji zavisi od rezolucije štampača ili ekrana. Ovde se adresiraju tačke, a jedinica mera je udaljenost između dve susedne tačke. Na primer, za laserske printere od 300 dpi (*Dots per Inch* – tačaka po incu) to je 1/300 inča, odnosno oko 0.085 mm. Neke izlazne jedinice, prvenstveno ekrani, nemaju istu horizontalnu i vertikalnu rezoluciju. Na primer, kod EGA je rezolucija 96x72 dpi.

Prilagođavanje slova izlaznoj jedinici. Najteži zadatak pri definiciji slova je definisanje osnovnih karakteristika dizajna i obezbeđenje da se one ne izgube prilikom štampe. Tu se podrazumeva konzistentnost širine trupova, očuvanje razmaka slova, izbegavanje tačaka koje eventualno štrče na obodima slova i slično. Svaki format zapisa font-fajla koji pretenduje da bude ozbiljan mora da sadrži mehanizam kojim se rasterizatoru daju uputstva kako da to izvede. U *PostScript* formatu to je složen mehanizam hintova. *TrueType* format ima još složeniji set instrukcija.

RASTERIZACIJA KONTURA

Rasterizacija je u suštini logički jednostavan proces: prema datom opisu izvuku se konture, a zatim popuni njihova unutrašnjost. Ali, da li je baš tako?

Slovo može da bude opisano sa jednom ili sa više zatvorenih kontura. Konture mogu da budu bilo zasebne bilo ugnježdene jedna unutar druge ili drugih, odnosno konture mogu da budu spoljne i unutrašnje. Dalje, i spoljne konture mogu da budu ugnježdene unutar drugih, što je čest slučaj kod raznih logotipa. Ono po čemu se razlikuju spoljne kontura od unutrašnjih je deo slova koji se popunjava: kod spoljnih se popunjava prostor unutar njih, a kod unutrašnjih prostor izvan konture. Na slikama 2a, 2b i 2c spoljne konture obeležene su sa „S", a unutrašnje sa „U".

Da li je kontura spoljna ili unutrašnja utvrđuje se na osnovu broja drugih kontura koje se nalaze oko nje. Ako je taj broj paran ili nula, kontura je spoljna. U suprotnom je unutrašnja. U skladu sa ovim, tačka oko koje je neparan broj kontura leži u delu like slova koji se popunjava, pa je treba prikazati. Tačka oko koje je paran broj kontura ne prikazuje se jer leži u delu koji se ne popunjava. Ovakav način popunjavanja lika slova poznat je kao pravilo par-nepar (*Even Odd Rule*) za popunjavanje.

Po pravilu, konture ne bi trebalo da se međusobno seku, ali mogući su izuzeci. Kako postupiti u tim slučajevima? Deo konture koja se seče sa drugom konturom je unutrašnja, a deo koji se ne seče spoljna kontura, pa bi po pravilu par-nepar deo konture bio popunjen, a deo prazan. Ali, da li je dizajner baš tako zamislio? Program koji vrši rasterizaciju ne može da „gleda u pasulj", pa je bolje ako mu se pruži dodatna informacija. To je rešeno usmeravanjem kontura i pravilom NZWR (*Non Zero Winding Rule*, slobodno prevedeno „raznosmerno pravilo"). Po ovom pravilu, deo gde se dve konture preklapaju biće popunjen ukoliko obe konture imaju isti smer (slika 3a). U suprotnom, taj deo će ostati nepopunjen (slika 3b). Smer konture određuje redosled koordinata u opisu, i može da odgovara ili da bude suprotan smeru kretanja kazaljke na satu.

Da bi se olakšao posao rasterizatoru, pojedini formati uvode dodatne konvencije. Na primer, kod *PostScript Type 1* formata, ako se prati smer konture, popunjeni deo uvek se nalazi sa leve strane. Kod *TrueType* formata je obrnuto: popunjeni deo je sa desne strane. Ukoliko se dva popunjena dela preklapaju, zajednička površina se ne popunjava. Ukoliko je deo te zajedničke površine unutar nekog trećeg popunjenog dela, on se opet popunjava. Dolazimo

do neke visle pai-nepai pravila dve popunjene površine daju nepopunjenu (za one kojima programiranje nije strano kraće objašnjenje je tačke na zajedničkim površinama se jednostavno XOR uju)

Posmatrajmo sad pojedinačne tačke Recimo da je potrebno nacrtati slovo u *em*-kvadratu čija je granulisanost daleko manja od originalne U takvim slučajevima veliki broj tačaka leži delimično unutar a delimično izvan konture Koje od tih tačaka popuniti a koje ne? Po logici stvari trebalo bi izvršiti zaokruživanje i popuniti one tačke koje veći deo imaju unutar konture odnosno čiji je centar unutar konture Ali ne lezi vraže U slučaju prikazanom na slici 4 na nekim mestima rastojanja između konture su veoma mala čak manja od razmaka dve susedne tačke Opet po logici stvari na tim deonicama došlo bi do ispuštanja tačaka i kidanja slika slova (*dropouts*) Suprotno centri nekih susednih tačaka „za dlaku" padaju unutar površina koje se popunjavaju Ako se obe tačke aktiviraju verovatno će doći do suvišnog zacrnjenja (**over-blackening**) U oba slučaja slovo gubi svoje karakteristike

Sledeći problem je prikazan na slici 5 Vidimo da aktiviranje samo onih tačaka čiji je centar unutar konture dovodi do mnogih nepravilnosti desni trup slova **m** je uži od ostala dva desna belina je veća od leve a levi i srednji serif se stapaju Primećujemo da je gornji serif deblji od donjeg pa čak i da svi serifi uz baznu liniju nisu iste debljine Mnogo grešaka na samo jednom slovu!

Tipičan problem jeste i postizanje simetričnosti krivih delova konture pošto kriva može samo malo da pređe preko sken linije ili da se savija neposredno pre sken linije To dovodi do suviše kratkih („kvigavih") ili suviše dugih prevoja, poput onih na slikama levog i srednjeg dela na slici 6 Popunjenje u desnom delu već je bolje

Još je mnogo elemenata od kojih zavisi kvalitet izgleda štampanog teksta kriva linija pod uglom od 45 135 225 ili 315 stepeni serifi izolovane tačke (one koje nemaju lateralni spoj sa tačkom iste boje) itd o čemu ovde neće biti govora

Ne treba međutim propustiti da se skrene pažnja na još jedan fenomen vezan za manu oka Slova koja su na dnu ili pri vrhu obla imaju veoma malo tačaka na donjoj i gornjoj bazi Ako bi najniža tačka takvog slova ležala tačno na baznoj liniji stekao bi se utisak da je pomereno malo naviše Ako bi pak najviša tačka ležala na liniji visine (malog ili velikog) slova obla slova bi izgledala niža od ravnih Zbog toga dno oblih slova treba da bude malo ispod bazne linije a vrh malo iznad linije koja označava visine slova u fontu kao što se vidi na slici 7



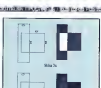
Slika 2a



Slika 2b



Slika 2c



Slika 3a



Slika 3b

HINTOVI

Pretpostavljam da oni koji još uvek čitaju ovaj članak imaju manje ili veće iskustvo sa programima koji omogućavaju dizajn slova, poput *CorelDraw!* ili *Publisher's Type Foundry* Ovi programi međutim ne omogućavaju definisanje hintova Istina *PTF* ima kao opciju automatsko hintovanje ali rezultati potvrđuju već iznetu konstataciju da u ovoj oblasti ništa ne može da nadmaši ljudsku ruku

Šta su hintovi?

Pošto je dobijen visokokvalitetan konturni opis dizajn mora da bude prilagođen da bi zadovoljio različite rezolucije i različite punktaže Sa smanjenjem veličine slova, a time i količine raspoloživih tačaka za njegovo crtanje raste potreba za modifikacijama i odstupanjima od originalnog oblika Upravo smo videli neke od problema koji se mogu javiti u procesu rasterizacije Dizajner treba da obezbedi instrukcije rasterizatoru kako da postupa kod kritičnih delova slova Takve instrukcije popularno se nazivaju *hintovi*.

Hintovima se kontrolišu modifikacije originalnog izgleda slova u cilju očuvanja karakteristika pisma Ove modifikacije možemo podeliti u tri grupe

1) Poravnanje (grid-fitting)

- podešavanja visine i širine slova kao i razmaka između slova
- premeštanje kontura da bi se izbegli prekidi (*dropouts*)
- proširenje ili suženje konture da bi se neutralisala izobličenja lika slova izazvana premeštanjem kontura
- izmene u cilju očuvanja simetrije i finoće krivih delova

2) Regularizacija

- izjednačavanje širine trupova
- neutralizacija optičkih varki prividnih zadebljanja na oštrim spojevima položaja okruglih slova i sl
- izbegavanje gubitka kontrasta između debelih i tankih trupova na minimalnim punktažama
- dodatna izobličenja lika u cilju očuvanja jednoobraznih težina krivih i dijagonalnih delova u odnosu na horizontalne i vertikalne

3) Očuvanje čitljivosti pri malim punktažama

- povećanje malih slova u odnosu na velika
- povećanje prevojnih delova
- proširenje slova

Za korektno definisanje hintova preduslov je dobro poznavanje formata u kome se font čuva kao i osobina programa koji treba da obradi font prilikom štampe O specifičnostima pojedinih formata biće reči u nekom drugom članku ukoliko se bude pokazalo da ima interesovanja za tako nešto Za sada ću se zadovoljiti da nabrojim metode hintovanja u nekim poznatijim formatima.

Compugraphic metod migracija kontrolnih tačaka *Compugraphic* je patentirao način prilagođavanja kontura određenim rezolucijama putem migracije kontrolnih tačaka ili ka najbližem centru tačke ili ka sredini između tačaka i to u horizontalnom vertikalnom ili u oba smera Migracija jedne tačke može da utiče na drugu tačku da je sledi linearno ili proporcionalno u odnosu na treću tačku Instrukcije su sadržane u dve liste hint pravila za vertikalne ivice i za horizontalne ivice Znači ne može se primeniti na dijagonale

Bitstream distribucija tačaka Bitstream je takođe patentirao svoj metod Najpre se odlučuje koliko redova tačaka pada u svaki deo unutar slova Zatim algoritam raspoređuje potreban broj tačaka u svaki deo konstrukcije po pravila po određenom prioritetu 1) zaokruženje širine na ceo broj 2) obezbeđenje razmaka među slovima 3) izjednačenje težine trupova 4) optičke korekcije 5) izjednačenje serifa 6) obezbeđenje razmaka između serifa 7) ostatak smestiti u centar Proces se primenjuje i vertikalno i horizontalno ali ne obezbeđuje i dijagonalnu kontrolu

URW Nimbus i Nimbus Q URW Nimbus metod je sličan tehnici koju primenjuje *Compugraphic* uz dodatak hintova za dijagonale i krive *Nimbus Q* podseća na *Bitstream* metod Kod oba načina javlja se problem očuvanja jednoobraznosti visina delova pojedinih slova

Adobe Type 1 Adobe obezbeđuje metod za kontrolu debljine horizontalnih i vertikalnih trupova kao i za očuvanje kontrasta između debelih i tankih trupova Uz to postoje mehanizmi za očuvanje jednoobraznosti celog fonta kao što su tipične linije poravnanja linije poravnanja oblih slova (*Blue Values*) tipična širina trupa i sl

Apple/Microsoft TrueType *TrueType* kao relativno nov format iskoristio je „vetar u leđa" i obuhvatio raznovrsne metode hintovanja počev od onih poznatih još u Knuth ovom *Metafont*-u pa preko proširenih *Compugraphic* i *Bitstream* metoda sve do tehnika iz konkurentskog *Type 1* formata Deluje impresivno što još uvek nije apsolutna garancija kvaliteta *TrueType* fontova Naime veoma je teško da se font kodira tako da koristi sve mogućnosti koje mu format pruža Eventualni korisnici treba da budu obazrivi jer će kupovinom *TrueType* fonta u 99 od 100 slučajeva dobiti običnu kopiju fonta koji već postoji u nekom drugom (čitaj Type 1) formatu Mnogo je praktičnije da kupe program za konverziju (preporučujem *FontMonger for Windows* verzija 1.0) koji će mu to uraditi „uživo" za mnogo manje para



Slika 4



Slika 5



Slika 6



Slika 7

VIŠE OD KNJIGOVOĐE

Vodite manje ili srednje preduzeće, ili ste njegov vlasnik, i želite da u svom poslu koristite i računar. Planirate da na taj način rešite administrativne obaveze i postignete veću poslovnost. Jedini problem je što ne znate odakle da počnete.

U poslednjih nekoliko godina kod nas je osnovan veliki broj firmi raznih profila. Uglavnom se bave trgovinom i uslugama, a nije mali broj ni onih koji se bave proizvodnjom. Ranije ili kasnije većina poželi da uvede računare u svakodnevnu upotrebu. Mnogi su iskoristili povoljne perioda za nabavku opreme kojih je i kod nas bilo, dok su drugi prvo stabilizovali svoje poslovanje, pa ovu ideju pokušavaju da sprovedu sa dušom u nešto nepovoljnijem trenutku. Ako tako razmišljate o svemu ovome i ste već našli i preduzeli, možda će vam dobro doći nekoliko saveta. Načelno se baš bavili izborom najboljeg računara ili idealnog programskog paketa koji sve radi sam. Želja nam je da vam pomognemo u sagledavanju celine problema i rešenja.

Računarska odiseja obično počinje onda kada budući korisnik, iznerviran gomilom ispreturanih papira na svom stolu (a nikako da utvrdi pravi promet prošlog meseca), pomisli kako bi mu dokumentacija bila ažurnija i preglednija da je formirana na računaru, koja se u početku sastoji samo iz računara i štampača i možda nekog programa. Nastaje idealan trenutak da se počne s razmišljanjem o svim činiocima koji utiču na srećan kraj priče.

Put koji je potrebno preći od ideje do realizacije je mračna prašuma, ali je dobro znati ga. Praktično gledano, ako se odlučite da automatizujete svoje poslovanje, potrebno je da preduzmete niz koraka koji se mogu ovako grupisati:

- definisanje poslova koji se obuhvataju obradom na računaru
- izbor opreme i programa
- priprema za uvođenje računara (šifriranje, prilagođavanje obrazaca)
- inicijalizacija (unos početnih podataka)
- eksploatacija

Prvo pitanje koje bitno utiče na realizaciju celog posla je šta se i *u kojoj meri* želi obuhvatiti obradom na računaru. Opšte primene računara, poput obrade teksta, obično nisu usko grlo u poslovanju, a njihova potreba uvek dođe na red negde na samom početku ili negde na kraju celog procesa, a ponekad je inicirana u ostatak sistema. Primena računara o kojoj ćemo nešto više govoriti, a koja je obično osnovni razlog za pokretanje procesa automatizacije, je vođenje nekog vida *knjigovodstva*. Činjenica je da je računar stvoren za tu vrstu posla i da je to jedna od najstarijih primena računarske tehnike uopšte, ali i da postoje varijacije u načinu na koji se to može uraditi. Pitanje mere u kojoj se poslovanje automatizuje, kao i pitanje pristupa realizaciji, određuju tok ostalih koraka u ovom procesu.

LINIJOM NAJMANJEG OTPORA

Ukoliko se računar želi uvesti u upotrebu sa što manje promena u postojećem načinu poslovanja firme, onda se primenjuje koncepcija nasleđena iz klasičnih računskih centara velikih firmi, koja je preovlađivala do pojave PC računara. Računar se u ovoj koncepciji koristi za naknadnu obradu podataka nastalih u poslovanju. Najčešće se ovakvim rešenjem obuhvata samo knjigovodstvena i računovodstvena evidencija. Originalna poslovna dokumentacija se formira ručno, odlaže se i zatim naknadno unosi i obrađuje na računaru. Rezultat ovakvog načina rada su izveštaji koji kasne za realnim vremenom, uglavnom rađeni da se zadovolji zakonom propisana dokumentacija (finansijsko knjigovodstvo, glavna knjiga) i izvrše naknadne analize poslovanja. Ovo je rešenje kojim se najmanje menja poslovanje firme, pa zato i ne nosi poseban rizik.

Dakle, ako ne želite potrese u svom načinu rada, a želite da zaposlite bar još jednog novog radnika koji će na računaru da obrađuje nastalu dokumentaciju, ovo je koncepcija koju treba da sledite.

Dejan Dinčić

SREDNJI PUT

Najčešća koncepcija danas je ona kojom se svi (ili bar većina) dokumenata koji nastaju u poslovanju firme formiraju na računaru. Prednost ovakvog sistema je da je mnogo veća ažurnost nego u prethodnom rešenju. U praksi to znači da isti radnici koji su posao radili ručno koriste sada računar da na njemu napišu otpremnicu, fakturu, nalog proizvodnji ili neki drugi dokument. Tako dokument nastaje istovremeno sa njegovim knjiženjem u svim potrebnim evidencijama. Jasno je da je ovim računar mnogo više „uvučen" u poslovanje firme i da time i poslovanje postaje zavisno od upotrebljene tehnologije. Ovo obično znači da je potrebno izvršiti reorganizaciju poslovanja, redizajnirati neke od obrazaca koji se koriste, izbaciti redundanse podataka i slično. Ovakve promene u većim firmama su obično deo projekta kojim se uvode računari u poslovanje, a u manjim firmama ponekad promakne važnost ovog elementa, pa su onda i promene bolnije.

POSTIĆI MAKSIMUM

Pored klasičnih knjigovodstvenih, komercijalnih ili proizvodnih podataka koji se obuhvataju prethodnom koncepcijom, računar se može još tešnje povezati sa poslovanjem firme. Primene su mnogobrojne, ali su sad već vrlo zavisne od oblasti u kojoj se koriste. Najčešće se radi o prikupljanju podataka u proizvodnim procesima, transportu i slično, njihovim povezivanjem sa centralnim delom sistema, a zatim i njihovoj upotrebi za upravljanje ovim procesima. Rešenja ove vrste su najskuplja i podrazumevaju najdublje izmene u postojećem načinu poslovanja.

Jasno je da se opisani nivoi zastupljenosti računara u poslovanju mogu postići i faznim uvođenjem, s tim što je tada potrebno imati viziju konačnog rešenja u samom početku, kako bi svaki osvojeni nivo mogao poslužiti kao osnova za primenu sledećeg.

Svakodnevnica koja nas okružuje i ograničenja koja nam nameće teraju nas da se u najvećem broju slučajeva okrenemo srednjem rešenju. S druge strane, i stepen razvoja mladih firmi je zapravo takav da se ovakvo rešenje čini najprimerenijim.

Šta od standardne evidencije, kojim redom i uz koje uslove može da se obradi na računaru?

Finansijsko knjigovodstvo je mesto na koje se slivaju svi ostali podaci. Ima veoma jasnu formu propisanu zakonom. Zapravo je to najstandardnija oblast poslovanja sa gledišta obrade na računaru. Njeno prebacivanje na računar ne bi trebalo da bitno utiče na tekući posao. Ipak, u slučaju iole većeg broja finansijskih transakcija potrebno je obaviti opsežan posao prenosa postojeće evidencije na računar.

Materijalno i robno knjigovodstvo u raznim formama su sledeći po zastupljenosti na računarima. U obradi na računaru ovo se knjigovodstvo obično proširuje u odnosu na tipično školsku formu pa obuhvata i druge elemente poslovanja.

Sledi niz primena računara u svakodnevnom poslovanju koje nisu direktno obuhvaćene knjigovodstvom: adresari, razne kalkulacije, obračuni kamata i slično. Neke od ovih funkcija proizvode podatke koji se koriste u knjigovodstvu, a druge su prosto operativno korisne.

Kada se odaberu koncepcija i funkcije koje će se automatizovati, potrebno je predvideti i stepen u kome će sve to biti međusobno povezano. Teorijski idealno rešenje bi zahtevalo maksimalnu povezanost funkcija (kako u poslovanju, tako i u programskom rešenju). U praksi to nije uvek moguće, a u neobičnim okolnostima pod kojima se odvija poslovanje naših firmi nije uvek ni potpuno upotrebljivo. Takođe, potreba da se funkcije sistema uvode postepeno tera da se omogući nezavisno korišćenje programskih celina. Kada se oprečni zahtevi pomire, proizlazi da treba predvideti rešenja koja omogućavaju tesnu vezu između poslovnih celina, ali ne obavezuju na nju. Na primeru to znači da finansijsko knjigovodstvo treba da ima mogućnost samostalnog rada, ali i mogućnost da automatizovano prihvata podatke iz robnog knjigovodstva.

KONFIGURISANJE SISTEMA

Na samom početku postavlja se pitanje: jedan, dva ili više nezavisnih ili mreža povezanih računara? Pored organizacionih i estetskih kriterijuma, odgovor je čvrsto uslovljen obimom podataka koji će se obrađivati. Postoji priličan broj manjih firmi čije se poslovanje može obuhvatiti jednim računarom. Takođe, ako se računarom obuhvataju potpuno raznorodni poslovi, moguće je koristiti dva odvojena računara. Na primer, jedan se može koristiti za pisanje pisama, faks poruka i slično, a drugi za knjigovodstvo i komercijalne podatke. U praksi već za dva, a naročito za više računara, povezivanje u lokalnu mrežu donosi brojne prednosti. U gornjem primeru i pisanje pisama i komercijalni dokumenti (koji su na različitim računarima) mogu koristiti isti adresar ako su računari povezani, pa je nemoguće da zabunom dođe do neslaganja podataka kao u slučaju dva odvojena adresara.

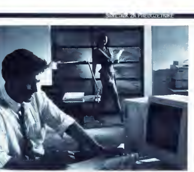
Osim što donosi prednosti, mreža je ponekad i jedina mogućnost. Čim količina podataka koja treba da se unese u računar prelazi fizičke mogućnosti jednog radnog mesta, jasno je da se mora predvideti mreža računara. Njome se omogućava unos podataka sa više unosnih mesta u jedinstvenu evidenciju, što je jedini način da se obuhvati takav obim podataka.

VIŠEKORISNIČKI SISTEMI

Kada je reč o upotrebi mreža računara, može zbuniti postojanje različitih standarda. Prvo, tu su DOS kompatibilne lokalne mreže (LAN). Radne stanice u takvoj mreži rade kao obični PC računari, s tim što mogu da koriste zajedničke resurse (obično jednog centralnog računara – servera). Tako se svi podaci nađu na disku servera, a može im se pristupiti sa više radnih stanica. Što se tiče izbora operativnog sistema mreže – kada se uzmu u obzir kvalitet i svi činioci naše stvarnosti (zastupljenost, podrška i sl.) – *Novell* mreže su bez premca. S obzirom na njihovu cenu, dužni smo da spomenemo i malu mogućnost da se izabere neki surogat. Naime, postoji nekoliko manje rasprostranjenih operativnih sistema slabijih karakteristika i skučenih mogućnosti, ali upotrebljivih za povezivanje manjeg broja računara. O njima se ipak treba dobro raspitati pre izbora.

Druga mogućnost za realizaciju više „računarskih" radnih mesta su višekorisnički operativni sistemi kao „XENIX" i „UNIX". Oni se od prethodnih razlikuju po tome što su radne stanice potpuno zavisni terminali (ekran i tastatura) priključeni na centralni računar. Još značajnije – razlikuju se i po tome što su znatno manje zastupljeni na ovim prostorima od DOS sistema, pa iako i za njih postoji ponuda poslovnih programskih paketa, mora se priznati da je izbor manji nego na polju DOS-a. Značajna prednost ovih sistema je jednostavnije uključivanje udaljenih poslovnica i njihovih terminala (telefonskom linijom) u centralni sistem. Na kraju, pitanje unixolik OS ili MS DOS mreža ne treba razrešavati na osnovu tehničke superiornosti jednog sistema na drugim, već na osnovu ukupnog utiska: čine ga širina ponude, kvalitet podrške, uslovi eksploatacije... što u većini slučajeva svodi izbor na DOS sisteme, izuzev kada specifičnosti poslovanja utiču na krajnji izbor.

O tehničkim karakteristikama opreme u *Računarima* se piše sasvim dovoljno. Ostaje nam samo da iznesemo neka praktična iskustva vezana za izbor i nabavku opreme.



- važno je obezbediti dobro održavanje – kvalitetan i postojan servis u slučaju kada vam je kompletno poslovanje vezano za računarsku opremu zastoj od par dana može da nanese ogromnu štetu poslovanju firme
- računari za poslovnu obradu podataka treba da imaju kvalitetne brze diskove ovo je element koji ima najveću težinu u ukupnom utisku neke konfiguracije dodatak memorije sa takođe može iskoristiti za ubrzanje operacija sa diskom upotrebom *cache* programe
- kvalitet štampača utiče na upotrebljivost celog sisteme pri tome je važnija njihova izdržljivost i pouzdanost od lepote ispisa i broja ukrasnih pisama kojima raspolažu
- redovan **backup** (pravljenje rezervnih kopija podataka) je uslov za bezbedno poslovanje uz upotrebu računara ne štedite na kvalitetu disketa ili strimer traka koja se za to koriste obavezno čuvajte više različitih kopija podataka
- ostavite razumnu rezervu za proširenje bilo da je u pitanju planirani kapacitet diska broj radnih stanica u mreži ili nešto treće
- upotreba standardne opreme garantuje vam da ćete lakše rešiti moguće probleme više je ljudi koji poznaju takvu opremu više i korisnika koji su se već sreli sa problemima diskovi koji se upotrebljavaju u svemirskim brodovima jesu dobri ali teško će se naći neko da ga ugradi a još teže da ga eventualno popravi

PROGRAMI

Na našem softverskom tržištu mogu se naći svi standardni programi za poslovanje preduzeća. Od njih da pojasnimo moguća su dva pristupa nabavka gotovog programskog paketa ili izrade programa prema zahtevu. Puno je argumenata koji idu u prilog nabavci gotovog programa. Pre svega cena razvoja je (po pravilu) znatno veća od cene gotovog proizvoda. U gotovom programu za koji se zna da je instaliran na većem broju sistema već su otklonjene greške od kojih svaki program boluje u početnoj fazi eksploatacije. Izrada programa prema zahtevu je opravdana kada je u pitanju specifično poslovanje koje postojeći programi ne obrađuju na zadovoljavajući način pa i tada samo ako dobitak opravdava ulaganje. Treba imati u vidu da razvoj programa prema zahtevu izisku je znatno veće angažovanje korisnika u raznim fazama projekta. Dakle kada procenjujete u kojoj meri gotov proizvod odgovara vašim zahtevima treba da prespitate moguća sporna mesta. Često je jednostavnije promeniti detalj svog poslovanja pa čak i veći deo u organizaciji poslovanja (naravno ako promena povećava efikasnost ili kvalitet rada) nego se upuštati u dugotrajnu odiseju razvoja.

Postoje kriterijumi na osnovu kojih korisnik može i sam da vrednuje ponuđene programe a da zbog toga ne mora da uči programiranje. Prezentacija programa ili uvid u njegovu praktičnu upotrebu u nekoj firmi daju naravno mnogo objektivniju sliku o programu nego čitanje reklamnih materijala. Idealno bi bilo da posle detaljne prezentacije programa korisnik uzme i demonstracionu verziju i praktično isprobava njegova mogućnosti. Pored elementarne ispravnosti programa (da radi ono za šta je namenjen) obratite pažnju i na sledeće elemente

- **efikasnost i logička postavka** prilično složen parametar za procenu ali svakako treba alarmirati višestruko unošenje podataka nedovoljno jasne odnose među podacima i slično
- **ponašanje programa sa većim količinama podataka** većina demonstracija programa se vrši sa vrlo malom količinom podataka bilo bi korisno videti kako program funkcioniše sa vašim polugodišnjim prometom na primer
- **fleksibilnost** ukoliko su u programu fiksirani čak i podaci koji su podložni promenama (stope poreza, intervali i slično) bićete osuđeni da i za najmanju zakonsku promenu nabavljate novu verziju programa naravno ne može se očekivati da korisnik menja i osnovnu logiku programa
- **višekorisnički rad** ako program podržava rad u mreži onda ne sme da nameće previše ograničenja za istovremeni pristup podacima
- **obezbeđenje podataka** zlo nikad ne spava pa je potrebno da program poseduje razrađen mehanizam za pravljenje zaštitnih kopija podataka kao i za njihovu restauraciju u slučaju da dođe do većeg poremećaja u radu (kvar i slično)
- **korisnički interfejs** izgled i manipulacija programom treba da budu što jednostavniji prirodniji. Iako je izgled subjektivna kategorija broj tastera koji je potrebno pritisnuti da bi se izvršile najčešće funkcije programa je nešto egzaktniji pokazatelj s tim što je nepoželjna krajnost u kojoj se funkcije pozivaju složenim kombinacijama tastera (Alt+Ctrl+Shift+Fn+) uz minimum tekstualnih objašnjenja
- **podrška** za programsku opremu je takođe važno da proizvođač može da obezbedi stručnu pomoć kao i nove verzije programa

PRIPREMA

Priprema za uvođenje računara u poslovanje zavisi od obuhvaćenih poslova i odabrane programske opreme. U većini slučajeva potrebno je kao jedan od prvih koraka preduzeti šifriranje podataka. Šifriranje ovde znači da je matičnim podacima kakvi su adresar i spiskovi artikala potrebno dodati oznaku koja ih jednoznačno određuje. Postoji nekoliko sistema za šifriranja. Uglavnom se razlikuju po tome da li šifra služi samo kao jedinstvena identifikacija ili treba da služi i za neku višu strukturiranje podataka. U prvom slučaju, na primer šifra artikla može biti prosto jedinstven broj koji se dodeljuje svakom novom artiklu prilikom dodavanja u bazu. U drugom slučaju šifra se može podeliti na nekoliko delova tako da prvi deo šifre označava robnu grupu kojoj artikel pripada drugi deo podgrupu a treći na primer redni broj artikla u okviru grupe. Ove složenije varijanta se obično primenjuje na većim skupovima podataka. Bez obzira koji se sistem šifriranja zabere on treba da omogući jednoznačnost i mogućnost prostog dodavanja novih šifara. Takođe treba mati u vidu na kojim će se sve mestima u programu ili budućim fazama automatizacije šifra koristiti tako da izabrani šifarski sistem zadovolji sve potrebe.

Za normalnu upotrebu računara u poslovanju potrebno je da se na njemu uspostavi ažurno stanje svih evidencija. Sa izuzetkom situacije u kojoj uvođenje računara prati početak rada firme to znači da je potrebno uneti neko početno stanje u programe kao i sve promene nastale od trenutka koji je obuhvaćen početnim stanjem do trenutka uključenja računara. Ovo pak najčešće znači da je potrebno sređivanje klasične „papirnate" evidencije da bi se dobile ispravne početne vrednosti za unos u računar.

U firmama sa manjim obimom podataka priprema je jednostavnija. Ali u većim preduzećima gde je zbog obima posao raspoređen na nekoliko službi faza pripreme zahteva ozbiljno planiranje i koordinaciju. Čak i uz optimalno ispunjene ostale zahteve ova faza može da stvori ozbiljne prepreke za normalno funkcionisanje novog sistema. Obuka zaposlenih za rad na novom sistemu spada u obavezni deo pripreme bez obzira na profil firme.

POČETAK

Iskustva u uvođenju računara u poslovanje su veoma raznolike. Mnogo je faktora koji utiču na krajnji efekat. Ipak i kod nas sada već postoji dovoljno akumuliranog iskustva pa se mnogi problemi mogu predvideti i izbeći.

Problemi mogu da nastanu u samom početku i to može da obeshrabri. Nakon svih priprema ili paralelno sa njima počinje unos početnog stanja. Ovaj posao se ne sme potceniti jer neuspostavljeno početno stanje može onemogućiti normalno funkcionisanje sistema. Ukoliko je obim podataka veliki potrebno je ponekad dodatno angažovanje kako bi se ovaj posao obavio dovoljno brzo i uspešno i računar mogao koristiti za obavljanje svakodnevnog posla. Tek kad računarski sistem uđe u normalnu eksploataciju korisnici mogu da sagledaju pun efekat automatizacije i osete zadovoljstvo što sistem funkcioniše. Ukoliko programi to omogućavaju pažljivom organizacijom može se rasporediti posao tako da paralelno teče unos početnog stanja i tekućeg posla.

U većim firmama važnu ulogu igra i reakcija zaposlenih koji se prvi put sreću sa računarom. Reakcije se kreću od potpunog prihvatanja (retko) do skepse ili potpuno iracionalne odbojnosti ili osećanja ugroženosti. Ovo se može loše odraziti na funkcionisanje sistema. Zato je potrebno obratiti pažnju na način na koji se prezentira nova tehnologija i u organizaciji jasno odrediti odgovornost zaposlenih za segmente posla koji zavise od njih. Tako se mogu umanjiti eventualni negativni efekti ličnog odnosa zaposlenih prema promenama.

EKSPLOATACIJA

Eksploatacija sistema je vreme kada su uglavnom prevaziđeni svi početni problemi. Novi način rada više nije nepoznanica. Pokazuju se efekti učinjenog izbora, e povremeno se javlja potreba da se neki novi delovi poslovanja preloće u računarsku organizaciju podataka. Od početka treba održavati disciplinu eksploatacije kako u pogledu ažurnosti tako u pomoćnim funkcijama programa. Ukoliko *Backup* ili pravljenje rezervnih kopija podataka nije čvrsto vezan za program treba da uvek postoji više različitih kopija podataka. Ispravnost kopija takođe treba povremeno proveravati.

Bez obzira na postignut nivo automatizacije gotovo da se uvek može i više. Pri tome treba što realnije odmeriti potrebe i ritam inovacija koje neće postati same sebi cilj.

SA C-a NA C++ U PET KORAKA

Ako postoji neko ko je zaslužan za to što se bavim programiranjem, onda je to sigurno Dejan Ristanović. Pre nekih pet-šest godina imao sam "komodor 128" i kuburio sa njegovim nepopravljivo sporim bejzikom. Tada su "Računari" objavili Dejanov umetak "Sa BASIC-a na Pascal" i ja sam prešao u divni svet struktuiranog programiranja. Moglo bi se reći da je ovaj tekst svojevrsno vraćanje duga.

Dejan Jelović

Već se dugo vremena bavim idejom da objektne jezike ne treba objašnjavati klasičnim "Hello world" pristupom. Naime, klasični programi se sastoje od naredbi i ništa nije logičnije nego krenuti od naredbe za ispisivanje teksta na ekran. Objektni jezici, s druge strane, nude još jedan nivo između naredbi i programa – klase. Pođimo zato od njih.

PRVI KORAK: KLASE

Osnovna sintaksna jedinica za C++ (moj drug Miljan bi rekao "inkrement") jeste klasa. Klasa nije ništa drugo nego malo poboljšana struktura (**struct**) preuzeta iz C-a. I dok u C-u pišemo

```
struct riba {
    int godine;
    float zarada;
};
```

odgovarajuća C++ konstrukcija bi izgledala ovako

```
class riba {
    int godine;
    float zarada;
};
```

Podacima koji se nalaze u klasi pristupa se potpuno isto kao i podacima u strukturi, pomoću operatora . i -> (tačka i strelica). Ono što C++ donosi kao novost jeste to da klasa, za razliku od strukture, može da sadrži funkcije.

```
class riba {
    int godine;
    float zarada;
    void print (void);
};
```

Naravno, kao i svaku "normalnu" funkciju koju deklarišemo, i funkciju deklarisanu u klasi moramo kasnije i definisati. Što se radi na sledeći način

```
void riba::print (void)
{
    printf ("Ona ima %d godina i zaradjuje %f
mesecno\n", godine, zarada);
}
```

Ovu funkciju možemo koristiti na sledeći način

```
riba sindi;
sindi.godine = 24;
sindi.zarada = 100000;
sindi.print ();
```

što daje

Ona ima 24 godina i zaradjuje 100000 mesecno

Kao što se iz primera vidi, funkcija koja pripada klasi definiše se sa <**ime klase**>::<**ime funkcije**>, u našem primeru **riba::print.** To što funkcija pripada klasi ima dve zgodne posledice.

1. Funkcija ima slobodan pristup svim podacima koji pripadaju klasi, pa tako funkcija **riba::print** iz primera slobodno koristi podatke **godine** i **zarada** koji pripadaju klasi **riba**.

2. Različite klase mogu da imaju funkcije sa istim imenom. To, prevedeno na srpski, znači da sada možemo da deklarišemo neku drugu klasu, recimo **drugarica**, i da u nju strpamo funkciju **print.** Kako prvu od ovih deklarišemo sa **riba::print**, a drugu sa **drugarica::print**, kompajler ne biva zbunjen i sve je O.K.

(Kad smo kod toga, C++ vam dopušta da i u istoj klasi definišete funkcije istog imena, ali sa različitim argumentima. Što znači, sasvim je legalno da imate **print (void)** i **print (int k)** u istoj klasi. Kompajler će znati da ih razlikuje po tome što jednoj prosleđujemo **void**, a drugoj **int.**)

Kako je ideja objektnog programiranja u tome da se prave biblioteke klasa, morao se naći način na koji bi se kontrolisalo ko koje podatke u klasi sme da dira. Zato su uvedene službene reči **public** i **private** koje određuju koji delovi klase su javni, a koje delove mogu da koriste samo funkcije koje su unutar klase. Recimo, u deklaraciji

```
class macka {
    private:
    int dobra_lovina;

    public:
    int godine;
    float zarada;
    void print (void);
    void podesi (void);
    int za_udaju (void);
};
```

polje **dobra_lovina** je privatno, dok su **godina** i **zarada** javni. I dok javnim promenljivima neke klase možemo bez problema da pristupamo kao na sledećem primeru, promenljiva **dobra_lovina** koja je privatna nije direktno dostupna. Važno je znati da je po definiciji sve što je unutar klase privatno, i da tek stavljanjem službene reči **public:** mi možemo da proglasimo neke delove javnim.

```
macka sindi;
sindi.godine = 24;
sindi.zarada = 1000000;
```

Pošto promenljivoj **dobra_lovina** ne možemo da pristupimo direktno, jedini način jeste da to uradimo preko funkcija koje pripadaju istoj klasi. Znači

```
void macka::podesi (void)
{
    dobra_lovina = (godine > 20) && (godine <
30) && (zarada > 100000);
}
int macka::za_udaju (void)
{
    return dobra_lovina;
}
```

Čemu sve ovo služi? Jednostavno, to je samo nov način organizacije programa. C++ je u osnovi još uvek C! Kao što ste iz prethodnih primera videli, jedina je razlika u deklaracijama – štampanje još uvek izvodimo sa **printf**, parametri funkcije su isti, *sve je isto!* Jedina novost je, za sad, to što i funkcije i podatke možemo da strpamo u jednu strukturu – klasu!

To je zgodno: stavljajući podatke i funkcije u jednu klasu, mi stvaramo jednu celinu. Ovaj mehanizam dosta podseća na izdvajanje funkcije i podataka koji bi činili celinu u jednu *.C datoteku i kasnije zajedničko povezivanje (link) u program. Prednosti klase u odnosu na datoteku kao osnovnu komponentu programa videćemo već u sledećem podnaslovu.

(Usput, nemojte se ljutiti za povremene izlete u muški šovinizam u mojim primerima. Kao što reče Knut, "autor ne misli baš sve tako".)

DRUGI KORAK: NASLEĐIVANJE

Da ostanemo na našem primeru klase **macka**. Recimo da se desi velika nesreća: devojke koje ste predstavljali tom klasom počnu da se udaju. Ne samo da je to samo po sebi loše, nego su i u vašem programu sada potrebne izmene. Treba napraviti novu klasu **udata_devojka** i u svakom delu programa koji je radio sa klasom **macka** napraviti još jedan koji bi radio sa klasom **udata_devojka**. Koliko je ovo neproduktivno, nema potrebe naglašavati. C++ zato nudi mehanizam kojim se svaka klasa može bezbolno proširiti. Recimo da želimo da u klasi **udata_devojka** bude klasa macka – informacija koliko je već udata – nova naredba **print** koja bi pored zarada i godina govorila i koliko je već udata. To bi se uradilo ovako

```
class udata_devojka: public macka
    public:
    int koliko_dugo;
    void print (void);
};
```

Naravno, kao i prošli put, moramo da definišemo i novu funkciju **print.** Recimo

```
void udata_devojka::print (void)
{
    printf ("Udata je vec %d godina, a ima %d
godina \n", koliko_dugo, godine);
}
```

Ovde se odmah vidi da nova funkcija **print** raspolaže i promenljivom **koliko_dugo**, s kojom je u istoj klasi, i promenljivom **godina** koja je u klasi **macka** koju je **udata_devojka** nasledila. (Ovde, do duše, treba paziti na službenu reč **private**. Sve što je privatno nije dostupno čak ni nasledniku. Tako da treba paziti da neku promenljivu koja će vam kasnije trebati ne strpate u privatni deo klase!)

Sada negde u programu možemo napisati

```
macka ana;
udata_devojka marija;

ana.godina = 19;
ana.zarada = 20000.00;
marija.godine = 20;
marija.zarada = 10000.00;
marija.koliko_dugo = 1;

ana.print ();
marija.print ();
```

što će ispisati

Ona ima 19 godina i zaradjuje 20000 mesecno
Udata je vec 1 godina, a ima 20 godina

Znači, za promenljivu **ana** biva pozvano **macka::print**, a za **marija udata_devojka::print**.

Funkcije mogu kao argument primati i klase, kao što mogu kao argument da primaju i strukture. Recimo, možemo imati funkciju **print_info** koja ima za zadatak da napiše informacije o promenljivoj iz klase **macka**.

```
void print_info (char *name, macka *ona)
{
    printf ("Ime: %s \n", name);
    ona->print ();
}
```

Ova funkcija bi, da joj prosledimo promenljivu **ana** iz prethodnog primera sa

print_info ("Ana", &ana);

dala

Ime: Ana
Ona ima 19 godina i zaradjuje 20000 mesecno

jer prvo ispisuje ime koje joj "normalno" prosleđujemo, a onda poziva i funkciju **print** klase koju je kao argument primila. Ono što je zanimljivo jeste to da funkciji **print_info** možemo da prosledimo i primerak klase **udata_devojka** koja je nasledila klasu **macka**:

```
udata_devojka marija;
marija.godine = 20;
marija.zarada = 10000.00;
marija.koliko_dugo = 1;
print_info ("Marija", &marija);
```

Rezultat ovog poziva je malo iznenađenje. Umesto da ispiše

Ime: Marija
Udata je vec 1 godina, a ima 20 godina

ispisaće

Ime: Marija
Ona ima 20 godine i zaradjuje 10000 mesecno

Šta se desilo? Pa, funkcija **print_info** je očekivala promenljivu klase **macka**, a dobila je promenljivu klase **udata_devojka**. Kako funkcija "ne zna" da smo joj prosledili primerak klase **udata_devojka**, ona je za podatke koje smo joj prosledili umesto da pozove funkciju **udata_devojka::print** pozvala **macka::print**!
Način da se ovo izbegne su takozvane virtuelne funkcije. Da smo u deklaracijama klasa **macka** i **udata_devojka** stavili ispred funkcija **print** reč **virtual**

```
class macka {
    private:
    int dobra_lovina;
    public:
    int godine;
    float zarada;
    virtual void print (void); /* !!!!!!!! */
    void podesi (void);
    int za_udaju (void);
};
class udata_devojka: public macka {
    public:
    int koliko_dugo;
    virtual void print (void); /* !!!!!!!! */
};
```

onda bi se svakom primerku klase **macka** dodao jedan pointer na funkciju **macka::print**, a svakom primerku klase **udata_devojka** pointer na funkciju **udata_devojka::print**. Sada se prilikom prenošenja klase kao argumenta nekoj funkciji prenosi i taj pointer, pa će sledeći primer

```
macka ana;
udata_devojka marija;
ana.godine = 19;
ana.zarada = 20000.00;
marija.godine = 20;
marija.zarada = 10000.00;
marija.koliko_dugo = 1;
print_info ("Ana", &ana);
print_info ("Marija", &marija);
```

dati

Ime: Ana
Ona ima 19 godina i zaradjuje 20000 mesecno



Ime: Marija
Udata je vec 1 godina, a ima 20 godina

tj. za svaku klasu biva pozvana baš njena funkcija **print**, a ne fiksna funkcija **macka::print**.

Predlažem da dobro savladate mehanizam virtuelnih funkcija jer on predstavlja suštinu objektnog programiranja. Veoma se često C++ programi prave tako što se napravi osnovna, tzv. apstraktna klasa sa svim virtuelnim funkcijama koje će nam trebati a koje ne rade ništa, a onda sve ostalo u programu pravimo tako da nasleđuje tu apstraktnu klasu i menja njene virtuelne funkcije.

Da se udaljimo od šovinističkih primera, recimo da nam treba program koji će da operiše sa vektorima i matricama. Definisali bismo jednu apstraktnu klasu sa svim potrebnim virtuelnim funkcijama, a onda bismo klase **matrica** i **vektor** izveli kao njene naslednike. Kada bismo sada definisali, recimo, uvezanu listu osnovne klase, u nju bismo bez problema mogli da smeštamo matrice i vektore, i još bolje, mogli bismo za bilo koji član te liste da pozovemo bilo koju njegovu funkciju koja je virtuelna! Na primer, ako je u osnovnoj klasi napravljena virtuelna funkcija **print**, mi bismo samo za svaki član liste pozvali funkciju **print** i razni vektori i matrice bi bili odštampani. Iako je meni bilo zgodno da ovakav mehanizam ilustrujem na vektorima i matricama, on se u principu može primeniti na bilo šta. No, ne treba zaboraviti da je C++ u isto vreme i C - nemojte gurati klase svuda već samo tamo gde za njima postoji realna potreba!

TREĆI KORAK: KONSTRUKTORI I DESTRUKTORI

C++ uvodi mogućnost stavljanja tzv. konstruktora i destruktora u klasu. To su u stvari sasvim obične funkcije, s tim što one bivaju pozvane automatski kada kreiramo promenljivu neke klase, i kada ta promenljiva izađe iz okvira u kome je aktivna. Šta to znači? Ako u funkciji deklarišemo promenljivu koja pripada nekoj klasi, automatski biva pozvan njen konstruktor, a po izlasku iz funkcije biva automatski pozvan njen destruktor. Kao konvencija, funkcija koja predstavlja konstruktor mora imati isto ime kao i klasa, a destruktor ima još znak tilda (~) ispred imena. Funkcije ne vraćaju ništa, čak ni **void**. Evo primera:

```
class primer {
    public:
    int i;
    primer (void);
    ~primer (void);
};
primer::primer (void)
{
    printf ("Pozvan je konstruktor \n");
    i = 5;
}
primer::~primer (void)
{
    printf ("Pozvan je destruktor, a i=%d \n", i);
}
```

U nekoj funkciji u programu možemo da stavimo

```
void neka_funkcija (void)
{
    int a, b, c;
    primer pera; // Biva pozvan konstruktor
    ...
    i = i * 2;
    ...
}                 // Biva pozvan destruktor
```

što će dati

Pozvan je konstruktor
...
Pozvan je destruktor, a i=10

Konstruktor takođe može da prima argumente i da na taj način uradi inicijalizaciju podataka pomoću naših podataka. Recimo da smo deklarisali klasu **primer** kao

```
class primer {
    public:
    int i;
    primer (int k);
    ~primer (void);
};
primer::primer (int k)
{
    printf ("Pozvan je konstruktor \n");
    i = k;
}
primer::~primer (void)
{
    printf ("Pozvan je destruktor, a i=%d \n", i);
}
```

sada bismo u programu umesto onog našeg

primer pera;

mogli da stavimo

primer pera (10);

i da time stavimo u promenljivu i vrednost 10.

Konstruktori u programima najčešće služe da bi se inicijalizovali podaci unutar klase, a destruktori da zajedno sa konstruktorima vrše alokaciju i oslobađanje memorije koja je potrebna za neke podatke klase. To znači da konstruktori često čine pozive funkcije **malloc**, a destruktori funkcije **free**.

ČETVRTI KORAK: PREKLAPANJE OPERATORA

Preklapanje operatora (...) operator (...) ... želite. Svaki izraz koji mu ... kompajler zapravo interno prevede u niz poziva funkcijama. Recimo, izraz 3 + 4 + 5 kompajler bi preveo u nešto poput

saberi (saberi (3, 4), 5)

I dok je ovaj mehanizam kod većine jezika "sakriven", C++ vam dopušta da ga koristite za klase ako treba. Svaki izraz tipa a+b kompajler prevodi u

a.operator+ (b)

gde je operator+ zapravi isto kao i bilo koja druga funkcija unutar klase kojoj a pripada. Na primer, klasu kompleksnih brojeva možemo definisati kao

```
class complex {
    public:
    float real, imag;
    complex operator + (complex c);
};
```

```
complex complex::operator + (complex c)
{
complex r;
    r.real = real + c.real;
    r.imag = imag + c.imag;
    return a;
}
```

pa onda kasnije u programu možemo pisati

```
    complex a, b, c;
    a.real = 10;
    a.imag = 20;
    b.real = 30;
    b.imag = 40;
    c = a + b;
```

pri čemu će kada program dođe do **a + b** u stvari biti pozvana funkcija **a.operator + (b)**, koja u ovom primeru sabira svoj realni deo i realni deo broja **b**, sabira svoj imaginarni deo i imaginarni deo broja **b**, i onda takav kompleksni broj vraća nazad.

Naravno, operator ne mora da bude preklopljen samo za jedan tip podataka. Isto kao što smo ovde preklopili operator + (**complex c**), tako u istoj klasi možemo preklopiti i operator + (**float f**) bez ikakvih problema, pa se kompleksni brojevi mogu lako sabirati sa realnim.

Ovo možda deluje malo komplikovano, sve dok se ne shvati da operatori nisu zapravo ništa drugo do obične funkcije neke klase. Naravno, kao što smo preklopili operator +, tako možemo da preklopimo i bilo koji drugi operator: *, <<, %, itd. Pošto su operatori u stvari obične funkcije, oni mogu biti nasleđivani, možete im u nekoj od narednih klasa promeniti funkciju, mogu čak biti i virtuelni. Jedini izuzetak među svim ovim operatorima jeste znak = koji ne može biti nasleđen. S obzirom na to da je ovo samo uvod u C++, za sada vam preporučujem da ga zaobilazite.

PETI KORAK: OSTALA PAPAZJANIJA

C++ ne samo da donosi objektna poboljšanja, on je u isto vreme i bolji C! Da počnem.

C++ omogućuje prenošenje parametara u funkciju po referenci. Za ljubitelje paskala, to je ono prenošenje argumenata sa **var**. Odgovarajuća deklaracija se razlikuje od obične za jedan znak &.

```
void neka_funkcija (int &i)
{
    ...
}
```

Sada se svaka promena promenljive i u ovoj funkciji odslikava na promenljivu sa kojom je funkcija pozvana. Nema više onog ružnog prenošenja adrese i operisanja sa pointerima unutar funkcije.

Moguće je funkciji odrediti **default** argumente. Recimo, deklaracija

```
void neka_funkcija (int i = 5)
{
    ...
}
```

znači da ovu funkciju možemo pozvati sa nekom vrednošću

```
neka_funkcija (20);
```

ili bez argumenta, u kom slučaju promenljivoj i biva dodeljen broj 5

```
neka_funkcija ();
```

Preklapanje funkcija. Kao i u ANSI C-u, svaka funkcija mora pre korišćenja biti bar deklarisana. Zbog toga je svaka funkcija tačno određena i njenim parametrima, pa je zato moguće imati više funkcija sa istim imenom ali različitim argumentima. Znači, sledeći segment je sasvim O.K.

```
void print (void)
{
    printf ("Ovo je neko stampanje \n");
}
void print (int i)
{
    printf ("Broj i = %d \n", i);
}
```

Sada u programu funkciju **print** možemo pozvati sa

```
print ();
```

ili sa

```
print (5)
```

a kompajler će znati na koju funkciju mislimo!

Promenljive se mogu deklarisati bilo gde u funkciji. Znači, umesto da deklarišemo sve promenljive na početku funkcije, sada ih možemo deklarisati baš tamo gde nam trebaju. Recimo

```
#include <stdio.h>
main ()
{
    printf ("Truc-truc \n");
    int i = 5;
    printf ("i = %d", i);
    return 0;
}
```

Vraćamo se nazad na ono "Hello world!". Kada na C-u želimo da izvedemo neko štampanje, mi za to koristimo zaglavlje **stdio.h** i funkciju **printf**. C++ nudi bolje rešenje: u zaglavlju **iostream.h** nalazi se deklaracija klasa **istream** i **ostream**, i njihovih primeraka **cin** i **cout**, respektivno. Ove dve klase imaju preklopljene operatore >> i << tako da za razne tipove podataka vrše ulaz/izlaz. Štampanje u C++ u bi se izvršilo recimo sa

```
cout << "Unet je broj " << 5;
```

a učitavanje sa

```
cin >> f >> i;
```

gde su **f** i **i** promenljive tipova **float** i **int**, respektivno. Ovo je urađeno tako što je za klasu **ostream** preklopljen operator << za razne tipove podataka a za klasu **istream** operator <<. Realizacija bi mogla da bude ovakva

```
class ostream {
    ...
    public:
    ostream operator << (int i);
    ostream operator << (float f);
};
ostream ostream::operator << (int i)
{
    printf ("%d", i);
    return *this; // Vrati samog sebe
}
ostream ostream::operator << (float f)
{
    printf ("%f", f);
    return *this;
}
```

(**This** je službena reč koja pokazuje na klasu.) U svakom slučaju, možete reći doviđenja ružnim **printf** i **scanf** formatima!

KLASNI PROBLEM

Pre nego što završim, dopustite mi da još jednom ponovim: C++ je u osnovi još uvek C. Sve novo što on donosi tiče se organizacije funkcija koje pozivate. S obzirom da smo ovde praktično samo prelteli preko glavnih osobina jezika, preporučujem vam neku dobru knjigu ili, ako ste baš uporni, helpove i primere Borlandovog C++ paketa. Ako je verovati Marksu, ova dva načina u učenju predstavljaju klasni problem. Baš me zanima da li se **Stroustrup** koji je napravio C++ sa svim tim klasama slaže?

NASTAVAK SA STRANE 59

va. Može se, na primer, pojaviti ako pokušate da izvadite koren iz negativnog broja ili kada referencirate registar u kome nema smislenog podatka.

Kontrolni registar (**control word register**) CW (slika 12) obezbeđuje prilagođavanje NDP-a potrebama korisnika. U upotrebi je samo 11 bita ovog registra, preostali su rezervisani za buduće generacije „Intel"-ovih aritmetičkih koprocesora.

O (infinity control) flag se koristi samo na 80286. Setovan stavlja koprocesor u takozvano *affine closure* stanje u kome postoje ne-brojevi plus i minus beskonačno. Ako je bit resetovan bit (*projective closure*) i preveliki i premali brojevi postaće „jedno isto" beskonačno. Koprocesor 80386 i NDP mikroprocesora 80486 uvek radi u *affine* stanju, pri čemu vrednost ovoga bita nije bitna; obično se zbog kompatibilnosti stavlja jedinica.

RC (rounding control) kontroliše zaokruživanje podataka. Obezbeđeni su metodi 00 (*round toward nearest*: zaokruživanje na najbliži broj, pri čemu se pri jednakim „rastojanjima" od dva broja zaokružuje na paran), 01 (*round up*: zaokruživanje „u pravcu" minus beskonačno), 10 (*round down*: zaokruživanje „u pravcu" plus beskonačno) i 11 (*round toward zero*: odsecanje viška cifara). Podrazumevano stanje je 00.

PC (precision control) određuje koji će se format racionalnih brojeva koristiti za smeštanje rezultata sabiranja, oduzimanja, množenja, deljenja i korenovanja. Obezbeđeni su formati 00 (*single precision* sa 24-bitnom mantisom), 10 (*double precision* sa 53-bitnom mantisom) i 11 (*extended precision* sa 64-bitnom mantisom). Vrednost 01 je rezervisana za buduće „Intel"-ove koprocesore. Ostale aritmetičke instrukcije uvek generišu *extended precision* rezultat, osim ako je samom instrukcijom određeno drugačije.

PM, UM, OM, ZM, DM i **IM** obezbeđuju maskiranje raznih vrsta grešaka na način koji smo već opisali. Recimo, OM = 1 nalaže NDP-u da ignoriše greške nastale usled prekoračenja (*overflow mask*).

Šesnaestobitni registar oznaka (**Tag word register**) omogućava NDP-u da u svakom trenutku „zna" šta se nalazi u kom registru aritmetičkog steka, na osnovu dva bita koja su tom registru „pridruženi". Vrednost 00 označava da je u registru smislen racionalan broj, 01 označava nulu, 10 obaveštava nas o beskonačnosti, *denormal* ili iz nekog drugog razloga neispravan podatak, dok vrednost 11 opisuje „prazan" registar sa slučajnim sadržajem. Programer u normalnim uslovima ne koristi niti (direktno) menja sadržaj ovog registra, ali će autori raznih debagera želeti da ga ispitaju kako bi ispravno prikazali stanje aritmetičkog steka.

Autore debagera će svakako interesovati i registri ukazatelja grešaka (**error pointer registers**) sa slike 13. Ovi registri se, doduše, ne mogu direktno čitati ni pisati ali ih instrukcija FSTENV kopira u memoriju. Njihov sadržaj se menja po izvršenju svake instrukcije koja operiše sa NDP-om, tako da će, ako instrukcija izazove interapt, na osnovu njih moći da se ustanovi gde je problem. Registri su neophodni pošto se rutina koja servisira interapt ne može osloniti na stanje „običnog" registra EIP – možda je, dok je izvršavanje aritmetičke instrukcije trajalo, glavni deo procesora izvršio još nekoliko drugih naredbi! Zato se stanje EIP-a pri pozivu instrukcije kopira u FIP (**floating-point instruction pointer**), CS se kopira u FCS (**floating-point code segment**), kod instrukcije u FOP (**floating-point opcode**), segmentna adresa operande (ako ga je bilo) u FOS (**floating-point operand segment**) a ofset operanda u FOO (**floating-point operand offset**). Prilikom analize ovih podataka treba obratiti posebnu pažnju na FOP – u njega se zapravo upisuje svega 11 bita koda instrukcije, pošto su sve instrukcije NDP-a kodirane tako da njihova binarna reprezentacija počinje sa 11011. Prvi primerci koprocesora 80387 nisu generisali ovaj podatak, niti je on na raspolaganju kada 80386 radi u zaštićenom režimu. Ukoliko, dakle, niste sigurni da će se vaš program izvršavati na 80486, preporučljivo je da kod instrukcije, uz neznatni gubitak na performansama, „pokupite" direktno iz memorije, sa adrese na koju ukazuju FCS i FIP.

U sledećim „Računarima" bavićemo se memorijom, tačnije načinom na koji 80486 koristi segmente i sprečava „neovlašćeno" posezanje jednog programa u prostor dodeljen nekom drugom.

Nucleus

dolazi !

Nucleus
Knjeginje Milice 10/I-37
YU-35000 Svetozarevo
Tel./Fax. (035) 224-629, 233-671

OSNOVE SORTIRANJA

Pošto smo se u prethodnim nastavcima upoznali sa osnovnim pojmovima iz teorije algoritama, krenućemo pravo u središte problema: sortiranje.

Bojan Petrović

Za početak smo izabrali smo neke osnovne algoritme za sortiranje podataka. Razloge za ovakav početak ima više: pre svega, proučavajući osnovne i najjednostavnije algoritme steći ćemo utisak o pravilima igre, naučili kako da razmišljamo pri rešavanju problema i primenili neke metode i tehnike koje će kasnije biti neophodne. Takođe, na ovaj način ćemo se lako upoznati sa terminologijom, a može se pokazati i da su ovi osnovni algoritmi sasvim dovoljni za naše potrebe. Konačno, zahvaljujući njihovoj "nadogradnosti" upoznaćemo se i sa nekim načinima poboljšanja performansi algoritama.

Kao što smo ranije pominjali, u više aplikacija se javlja situacija da se sortiranje vrši samo jednom (ili nekoliko puta) ili je broj elemenata koji se sortiraju mali, recimo manji od 100. Tada je sasvim logično izbeći neki žestok algoritam za čije testiranje i podešavanje potrošimo nekoliko desetina puta više vremena od samog njegovog rada. Za ovako mali broj elemenata (odnosno samo jedno izvršavanje) korisnik neće mnogo propatiti. Sa druge strane, neki daleko složeniji algoritam koga smo jednom pažljivo proučili (izgubivši ko zna koliko vremena) davaće impresivne rezultate za ogromne setove podataka, kada oni lošiji ne mogu ni da se porede. Zaključak je sledeći: mada možemo imati omiljeni algoritam koji zadovoljava naše potrebe u mnogim situacijama, uputnije je birati algoritam "po situaciji", odnosno znati koji algoritam i kada.

NEKOLIKO POJMOVA

Na početku upoznajmo se sa nekim pojmovima. Pre svega, pojam složenosti algoritma smo ranije već obradili. Podsetimo se, za nas je od velike važnosti vreme rada algoritma (odnosno broj izvršenih koraka) koje iskazujemo preko pomenute složenosti. Ako je jedan algoritam kvadratne složenosti, brži od njega će biti onaj linearne složenosti (mada se može pokazati da ni jedan algoritam za sortiranje u prosečnom slučaju ne može imati manje od **N*logN** poređenja).

Druga karakteristika je korišćenje memorijskih resursa. Ako se sortiranje vrši nad podacima u memoriji, govorimo o internoj metodi, a ako radimo sa podacima na disku, reč je o eksternom sortiranju (u ovom nastavku se bavimo nekim internim metodama). Dalje, obratićemo pažnju i na količinu memorije potrebnu za rad algoritma. Tu možemo napraviti još jednu malu klasifikaciju: algoritmi koji za sortiranje N elemenata ne koriste dodatne resurse (ne računajući stek), oni koji koriste manju količinu dodatne memorije i oni koji čine potpunu kopiju sortiranih elemenata na nekom mestu u memoriji. Cilj je, naravno, da obezbedimo što manju potrošnju memorijskih resursa.

U praksi se nekad javlja i potreba za stabilnošću algoritma. Ako pokušamo da po abecedi sortiranu listu studenata sortiramo po proseku ocena, mogu se (u osnovi) dogoditi dve stvari: nova lista će imati grupe studenata sa istim prosekom sortirane po abecedi ili će se prethodno sortiranje potpuno izgubiti. U prvom slučaju, kada se zadržava relativni odnos elemenata pri sortiranju, govorimo o stabilnom algoritmu.

Većinu algoritama ćemo isprobavati na nizu "**primer algoritama**". Razlozi za ovo su čisto praktične prirode. Pre svega, na ovaj način se potpuno ograđujemo od svih mogućih problema oko veličine zapisa, međusobnog upoređivanja, zamene mesta i slično. Uostalom, ako se i javi potreba za sortiranjem neke datoteke sa velikim zapisima, nećemo trošiti vreme na prebacivanje velikog zapisa sa jednog mesta na drugo, već ćemo koristiti poseban niz sa pokazivačima, kada ćemo pomeranjem pokazivača indirektno pomerati zapise. Takođe, pošto ćemo celobrojne vrednosti koristiti kao pokazivače, gornjim nizom ističemo razliku između niza koji se sortira i pokazivača koji se pri tom koristi.

SELECTION SORT

Jedan od najjednostavnijih algoritama za sortiranje je sledeći: pronađi najmanji element u nizu i zameni ga sa onim na prvom mestu, nađi zatim sledeći najmanji element i zameni ga sa onim na drugom mestu u nizu, nastavi na isti način dok ceo niz nije sortiran. Ovo rešenje je tipa "zdrav razum" i vrlo često ga programer sam pronađe rešavajući problem sortiranja prvi put (listing 1).

Opisana implementacija je prvi način da se izvede ovaj algoritam. U prvom koraku prvog prolaza se ustanovi da je "P" manje od "R", pa se ništa ne čini (naravno, posmatramo gore navedeni niz). U drugom koraku se nalazi na "I" i zamenjuje se "P". U trećem koraku se "M" preskače, a u četvrtom se "E" smešta na prvo mesto. Peti korak upoređuje "E" sa "R" (i preskače), i konačno u šestom koraku se nalazi na "A" koje stiže na svoje prvo mesto, dok "E" staje na sedmom mestu. Do kraja ovog kruga više neće biti premeštanja jer su svi elementi nadalje veći ili jednaki "A". Konačan rezultat prvog od n prolaza je "A" na prvom mestu, a ostatak niza blago izmešan. Drugi krug se odvija na sličan način, ali za drugu poziciju, i tako dalje. Iz ovoga se odmah vidi nevolja: tražeći najmanji elemenat u nizu izvršili smo mnogo prebacivanja, što jede dosta vremena, naročito kod velikih zapisa.

Očigledno rešenje je da ne vršimo prebacivanja dok ne nađemo baš najmanji elemenat (listing 2).

Ovog puta premeštanje vršimo tek kada smo sa sigurnošću utvrdili da je u i-tom prolazu nađen odgovarajući i-ti najmenji elemenat u nizu. Dakle, za svako i (gde se i menja od jedan do n-1) zamenimo **a[i]** sa najmanjim elementom u **a[i]..a[n]**. Kako i putuje na desno, tako za sobom ostavlja deo niza u svom konačnom izgledu (čiji elementi više neće biti dirani). Slika jedan ilustruje upravo rad ove druge (bolje) implementacije algoritma.

Probajmo sada da izračunamo red složenosti ove metode. Za svako i (od **1** do **n-1**) imamo jedno prebacivanje i **n-i** upoređivanja, što ukupno daje **(n-1)+(n-2)+...+2+1 = n(n-1)/2** upoređivanja i **n-1** prebacivanje. Očigledno, pošto je kvadratne složenosti, algoritam je jako osetljiv na broj podataka koji se obrađuju, te je dobar za manje datoteke. Drugo važno zapažanje je da se **i**-ti elemenat samo jednom pomera, pa je *Selection sort* dobar za datoteke sa velikim zapisima. Konačno, primetimo i da ova metoda ne zahteva dodatne memorijske resurse.

INSERTION SORT

Malo (baš malo) komplikovaniji algoritam od prethodnog jedan po jedan elemenat poredi sa već sortiranim elementima i ubacuje ga (**insert**) na odgovarajuće mesto (čuvajući sortiranost, naravno). Umetanje se vrši jednostavnim pomeranjem većih elemenata od našeg udesno. Implementacija je data na listingu 3.

Za svako **i** (gde se i menja od **2** do **n**) elementi od a[1]..a[i] se sortiraju umetanjem **a[i]** na odgovarajuće mesto u odnosu na već sortirane elemente a[1]..a[i-1]. Kako i putuje desno, tako se javlja neka uređena grupa elemenata, ali oni ne moraju biti u svojim konačnim pozicijama. Naravno, niz postaje sortiran kada i stigne do kraja. Rad algoritma je ilustrovan slikom 2.

Ako se malo bliže zagledamo u program, videćemo da postoji jedan bag. Naime, ako je a[i] najmanji element u nizu, **while** će pokušati da ga odgura levo od a[1]! Ovo se može rešiti na dva načina: ispitivanjem da li je **j**>1 i ubacivanjem granične vrednosti u **a[0]**. Prvi način znači da ćemo proveriti j vrlo stalno, mada se situacija j=1 javlja retko, i time jesti dobro parče vremena. S druge strane, granična vrednost (manja elemenat od najmanjeg u nizu) je u redu ako smo u stanju da definišemo tu vrednost.

Na sličan način kao kod *Selection* sorta, dolazimo do činjenice da ovaj algoritam troši oko **(n*n)/4** upoređivanja, odnosno **(n*n)/8** prebacivanja u proseku, što znači da je i ovaj algoritam kvadratni. Uostalom, većina zaključaka je ista kao kod prethodnog algoritma, pa se nećemo dalje zadržavati. Napomenimo samo jednu veoma važnu osobinu, do koje je moguće doći uz malo razmišljanja: *Insertion* je linearan za skoro sortirane datoteke.

BUBBLE SORT

O ovoj metodi samo kratko. Algoritam se svodi na sledeće: prolazi kroz datoteku upoređujući po dva elementa i izmenjujući ih ako je to potrebno. Prolaženje se obavlja ispočetka sve dok postoje dva elementa koja treba da izmene mesta. Pokušajmo ovo da shvatimo baš za najveći elemenat u datoteci: u prvom prolazu se on upoređuje redom sa svakim sledećim i

Listing 1

```
select (int a[], int n)
  {
    int i,j,temp;
    for (i=1; i<n; i++)
        for (j=i+1; j<=n; j++)
          if (a[j]<a[i])
             { temp=a[i]; a[i]=a[j]; a[j]=temp; }
  }
```

Selection sort

Listing 2

```
select(int a[], int n)
  {
    int i,j,min,temp;
    for (i=1; i<n; i++)
      {
        min=i;
        for (j=i+1; j<=n; j++)
          if (a[j]<a[min]) min=j;
        temp=a[min]; a[min]=a[i]; a[i]=temp;
      }
  }
```

Poboljšani Selection sort

Listing 3

```
insertion (int a[], int p[], int n)
{
  int i,j,v;
  for (i=0; i<=n; i++) p[i]=i;
  for (i=2; i<=n; i++)
    {
      v=p[i]; j=i;
      while (a[p[j-1]]>a[v])
        { p[j]=p[j-1]; j--; }
      p[j]=v;
    }
}
```

Insertion sort

Listing 4

```
bubble (int a[], int n)
  {
    int i,j,temp;
    for (i=n; i>=1; i--)
      for (j=2; j<=i; j++)
        if (a[j-1]>a[j])
          { temp=a[j-1]; a[j-1]=a[j]; a[j]=temp; }
  }
```

Bubble sort

pomera za jedno mesto udesno, sve dok ne dođe na krajnju desnu poziciju. U drugom prolazu se isto događa sa drugim najvećim elementom i tako redom. Moguća implementacija je data nam listingu 4.

Naravno, kao i ranije, primetimo da je red složenosti ovog algoritma kvadratan, da ne troši dodatne memorijske resurse i da je pogodan samo za manje datoteke.

PRVI ZAKLJUČCI

Pokušajmo sada da uporedimo prethodna tri algoritma i izvučemo neke zaključke. Pre svega, primetili smo da su sva tri kvadratnog reda složenosti u proseku, da ni jedan ne zahteva dodatne resurse i da su jednostavni za implementaciju. U prvom tekstu serije smo već objasnili da to što su istog reda složenosti ne znači da će im vreme rada biti baš isto, nego će biti istog reda veličina! Razlike, dakle, treba tražiti u nekim situacijama vezanim za određene situacije.

Vratimo se na pojam "skoro" sortiranih datoteka. Tu smo mislili na one slučajeve kada je deo elemenata već sortiran i na svojim konačnim pozicijama, ili je nekoliko sortiranih "komada" datoteke međusobno nesortirano. Recimo, pri sređivanju potpuno sortiranog niza, *Insertion* će odmah za svaki element videti da je na svom mestu i preskočiti ga, što će uraditi i *Bubble*, pa će njihovo vreme rada u ovom ekstremnom slučaju biti linearno. *Selection* će se, pak, pošteno namučiti i dalje pokušavajući da pronađe najmanji element, odnosno i dalje će njegovo vreme rada biti kvadratno. Ova interesantna osobina će kasnije biti od koristi ako nađemo neki jači algoritam koji je u opštem slučaju brži od ovih (na primer **N*logN**), pustićemo ga da radi sve dok se niz ne sortira do nekog stepena kada možemo reći da je skoro sortiran. Tada skačemo na *Insertion* (linearan za skoro sortirane) i eto ušteda.

Pogledajmo i uticaj broja upoređivanja i prebacivanja na rad algoritma. Pre svega, prebacivanje dva elementa je gotovo uvek daleko duže od njihovog upoređivanja, a razlika će najviše zavisiti od veličine elemenata koji se pomeraju. Za veoma velike zapise pre svega, mnogo je značajnije da se zapisi što manje premeštaju tokom rada (u principu, bolje je da izvršimo dva tri upoređivanja nego jedno prebacivanje takvog zapisa). *Selection* će tada trošiti samo deo vremena u odnosu na *Bubble*, dok će *Insertion* biti negde između. U stvari, *Bubble* će gotovo uvek biti dva puta sporiji od *Insertion* sorta, bez obzira na uslove. Za mala zapise, prva dva metoda će biti veoma blizu po performansama, dok će treći opet daleko zaostajati.

Dakle, *Selection* treba koristiti za male datoteke sa velikim zapisima, a *Insertion* za male datoteke sa manjim ili malim zapisima koje su možda i skoro sortirane. Treći algoritam u poređenju sa ovima ostaje bez posla, ali ga je zato najlakše implementirati.

SHELL SORT

Ova metoda je prosto poboljšanje *Insertion* algoritma, nastala iz sledećeg primera: ako se najmanji elemenat niza nalazi na poslednjem (**n**-tom) mestu, biće potrebno n koraka da stigne na prvo. Dakle, cilj

Listing 5

```
shell (int a[], int n)
 {
  int i,j,h,v;
  for (h=1; h<=n/9; h=3*h+1);
  for ( ; h>0; h/=3 )
   for (i=h+1; i<=n; i+=h)
    {
     v=a[i]; j=i;
     while (j>h && a[j-h]>v)
      { a[j]=a[j-h]; j-=h; }
     a[j]=v;
    }
 }
```

Shell sort

Listing 6

```
quick (int a[], int l, int r)
 {
  int i;
  if (r>l)
   {
    i=podela(l,r);
    quick(a,l,i-1);
    quick(a,i+1,r);
   }
 }
```

Quick sort

Listing 7

```
quick (int a[], int l, int r)
 {
  int desni,i,j,temp;
  if (r>l)
   {
    desni=a[r]; i=l-1; j=r;
    for (;;)
     {
      while (a[++i]<desni);
      while (a[--j]>desni);
      if (i>=j) break;
      temp=a[i]; a[i]=a[j]; a[j]=temp;
     }
    temp=a[i]; a[i]=a[r]; a[r]=temp;
    quick(a,l,i-1);
    quick(a,i+1,r);
   }
 }
```

Potpuna implementacija quick sorta

ovog algoritma bi bio da ubrza prebacivanje, dozvoljavajući premeštanje elemenata koji su međusobno jako udaljeni.

Kako bi ovo funkcionisalo? U prvom prolazu poredimo prvi elemenat sa devetim, drugi sa desetim, treći sa jedanaestim i tako dalje (niz u ovom primeru očito ima 16 elemenata). U drugom prolazu idemo ponovo poredeći sad prvi sa petim, drugi sa šestim i tako dalje. Uzimajući veliko **h** (gde je **h** rastojanje među elementima) pomeramo elemente koji su na velikoj udaljenosti, čime olakšavamo **h** sortiranje elemenata za neko manje **h**. Koristeći ovakvu proceduru na sekvencu vrednosti **h** koja se završava sa jedan, dobijamo sortirani niz. Implementacija algoritma na listingu za **h** uzima vrednosti ... 1093 364 121 40 13 4 1 (listing 5).

Sekvenca brojeva koja se koristi za **h** je od velike važnosti. Postoje neke sekvence koje daju bolje rezultate od gornje, ali neke i lošije (možda bi bilo interesantno da čitalac za isti niz proba nekoliko različitih sekvenci brojeva). Recimo, sekvenca 64 32 16 8 2 1 nije dobra, jer se elementi na parnim i neparnim pozicijama upoređuju tek u zadnjem koraku. U nekim implementacijama *Shell* sorta bira se sekvenca počevši od h = n. Ovo može raditi dobro u nekim slučajevima, ali pre ili kasnije će se naći neko takvo n da sekvenca daje veoma loše rezultate. Ako vam se sve ovo činilo nepreciznim, to je zato što niko do sada nije uspeo stvarno da analizira algoritam. Za razne sekvence rad algoritma se menja, pa i njegov red. Za sekvencu koju smo mi izabrali red algoritma je **SQR(n*n*n)**, što je daleko bolje od **n*n** ranijih metoda. Kako se ovo stvarno izračunava je daleko od predmeta ovog teksta, pa nećemo ni pokušavati.

Ova metoda je obično prva na listi izbora za manje datoteke (recimo 5000 elemenata i manje), zahteva malo koda. U stvari, u kasnijim nastavcima, algoritmi koje budemo upoznavali će biti možda tek dva puta brži (i to u nekim situacijama) za razumno veliko **n**. Ukratko, ovo je algoritam koga se treba prvo latiti. Kasnije, ako se pokaže da je potrebno ubrzati sortiranje, može se pokušati sa nekim boljim (ali daleko komplikovanijim) algoritmom.

QUICK SORT

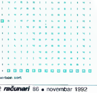
Konačno, stigli smo do finala. Algoritam koji je pred nama je verovatno najomiljeniji metod mnogih programera. Osnovnu ideju je dao C.A.R. Hoare 1960. godine i od tada je bio predmet mnogih studija i diskusija. Njegova popularnost leži u činjenici da je jednostavan za implementaciju, radi jako dobro gotovo u svim situacijama i troši malo resursa.

Quick sort algoritam spada u "podeli i pobedi" algoritme koje smo već pominjali ranije. Ideja je da se izvrši podela datoteke na dve, koje će se sortirati svaka za sebe (i onda rekurzija – listing 6).

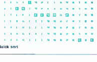
Parametri l i r su granice pod datoteka koju sortiramo u tom trenutku, što znači da se gornja procedura mora pozvati sa **quick(niz,1,n)**. Centar metode je procedura "podela" koja mora niz tako preurediti da sledeća tri zahteva budu ispunjena:

- element **a[i]** je na svom konačnom mestu u nizu za neko i
- svi elementi u **a[l]..a[i-1]** su manji ili jednaki elementu **a[i]**
- svi elementi u **a[i+1]..a[r]** su veći ili jednaki elementu **a[i]**.

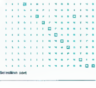
Ovo znači da slobodno biramo neki elemenat **a[r]** koji će se po završetku naći na svom konačnom mestu (element podele). Sada pretražujemo sa levog početka niza tražeći veći element, a sa desnog kraja manji. Dva pronađena elementa su očigledno na lošim mestima, pa ih međusobno zamenjujemo (u stvarnosti traže se veći-jednak i manji-jednak element). Nastavljajući ovako, obezbedićemo da svi elementi manji od **a[r]** budu u skladu sa zahtevom 2, a svi veći elementi u skladu sa zahtevom 3. Proces se, dakle, završava onda kada se pokazivači koji ma smo se kretali udesno i ulevo sretnu na istom elementu, što je **a[i]** iz gornjeg uslova. Tada zamenjujemo element **a[r]** sa njim (obično se smatra da je **a[i]** onaj na kom se zaustavio levi pokazivač). Sada je niz daleko sređeniji u odnosu na originalni i postupak se nastavlja za levi i desni podniz u odnosu na naš ele-



Insertion sort



Quick sort



Selection sort

Listing 8

```
#define max 100
static int stack[max+1],s;
push (int i)
  { stack[s++]=i; }
int pop()
  { return stack[--s]; }
stackinit()
  { s=0; }
int stackprazan ()
  { return (s); }

quick (int a[], int N)
  {
    int i,l,r;
    l=1; r=N; stackinit();
    for (;;)
      {
        while (r>l)
          {
            i=podela(l,r);
            if (i-l>r-i)
              { push(l); push(i-1); l=i+1; }
            else
              { push(i+1); push(r); r=i-1; }
          }
        if (stackprazan()) break;
        r=pop(); l=pop();
      }
  }
```

Implementacija quick sorta sa stekom umesto rekurzije

ment podela. Potpuna implementacija data je procedurom na listingu 7.

Ovde je **deobni** trenutna vrednost deobnog elementa, a pokazivači su j i l. Procedura **podela** je ovde beskonačna petlja (for (;;)) iz koje se u pogodnom trenutku iskače sa **break** (dakle, onda kada se pokazivači sretnu). Unutrašnja petlja ove implementacije je najobičnije inkrementiranje pokazivača i poređenje elemenata sa konstantom, što upravo i čini ovaj algoritam tako brzim. U prvom nastavku serije smo pričali o uticaju unutrešnje petlje na vreme izvršavanja algoritma i ovde smo se upravo sa tim slučajem susreli: pošto se ova petlja izvršava stalno iznova tokom rada algoritma, vreme njenog rada najviše utiče na dužinu rada cele procedure, naša unutrašnja petlja teško da može biti brža.

Kao i kod *insertion* sortiranja, i ovde se javlja problem ograničenja. Naime, ako je deobni elemenat najmanji u nizu, druga **while** petlja će pokušati da izleti iz niza. Ako je deobni elemenat najveći u nizu, problema nema (u našoj implementaciji), jer će prva **while** petlja naći baš na njega pre nego što izleti iz niza. Malo kasnije, videćemo kako da izbegnemo ovakve okolnosti.

Mali problem koji se ovde javlja je i činjenica da ovaj algoritam u nekim slučajevima radi sporije nego što bi trebalo. Za već sortirani niz, vreme rada skače opet na **n*n**, a to svakako nismo želeli. Srećom, postoje načini da se ovo predupredi u aplikaciji. Drugi problem su datoteke sa više istih elemenata. Tada se postavlja pitanje šta raditi kada pokazivači naiđu na elemente jednake deobnom: da li da oba pokazivača stanu, da li da stane samo jedan, ili da oba prođu? Matematički je pokazano (a mi to nećemo pokušavati) da je najbolje da oba pokazivača stanu. Sličan slučaj je stelanje pokazivača na elementu jednakom deobnom. Za ovu situaciju moguće je unaprediti našu implementaciju, ali to ostavljamo čitaocu kao vežbu.

Zaustavimo se na trenutak na performansama *Quick* sorta. Zgodno bi bilo pretpostaviti da se u svakoj deobi datoteka delila na dva jednaka dela. Tada broj upoređivanja iznosi (iz rekurzivnog izraza) **2N*lnN**. Kako je ovo zapravo prosečan slučaj, možemo sa velikim pouzdanjem reći da će vreme rada algoritma biti reda **N*logN (2N*lnN je približno 1.38N*logN)**.

Naravno, šta bi bilo od nas ako ne bismo težili poboljšanjima. Prvo što nam pada na pamet je uklanjanje rekurzije upotrebom steka, o čemu smo već govorili u prvom nastavku serije. **Implementacija** je data listingom 8.

Očigledno je kako smo implementirali stek, kao i da vrednost max = 100 mora biti podešena pravilno na vreme. Zahvaljujući **if-else** linku u implementaciji na stek idu samo podaci veće pod-datoteke, pa će ukupan broj ulaza na stek biti oko **logN** (svaki ulaz će predstavljati polovinu prethodne pod-datoteke). U rekurzivnoj verziji moglo bi se dogoditi da se na steku nađe i **n** elemenata (recimo za sortirani niz).

Listing 9

```
insertion (int a[], int n)
  {
    int i,j,v;
    for (i=2; i<=n; i++)
      {
        v=a[i]; j=i;
        while (a[j-1]>v)
          { a[j]=a[j-1]; j--; }
        a[j]=v;
      }
  }
```

Insertion metoda u sortiranju dugačkih zapisa

Drugo poboljšanje se bazira na sledećoj primedbi: ako obe pod-datoteke imaju po jedan elemenat, obe će biti postale na stek, da bismo odmah zatim bile skinute sa njega. Ustvari, ovo možemo proširiti i na male pod-datoteke. Jedan način bi bio da tada zovemo *insertion* sort, što bismo izveli zamenjujući uslov **if (r>l)** sa **if (r-l<=K) insertion(l,r)**, gde je **K** parametar koji kaže za koju veličinu pod-datoteke se skače na *insertion*. Ovo u većini slučajeva daje oko 20% poboljšanja. Nešto lakše bi se to moglo izvesti jednostavno (u rekurzivnoj verziji) zamenjujući **if (r>l)** sa **if (r-l>K)**. Tada potpuno preskačemo uređivanje malih pod-datoteka, a na kraju dobijemo skoro sortiranu datoteku. I to je ideja, već smo rekli da je za ta slučajeve *insertion* najbolji!

Treće poboljšanje bismo mogli da izvedemo menjajući način uzimanja deobnog elementa. Opet dva načina: prvi je biranje slučajnog elementa iz datog niza, a drugi se naziva **srednji-od-tri**. U prvom slučaju se samo izbegava najgori slučaj (već pominjan), ali se on i tako retko dešava. Druga metoda je bolja u opštem slučaju: uzimamo tri elementa iz niza, i to redom s kraja, sredine i početka, sortiramo ih i uzimamo srednji od njih, njega zamenjujemo sa **a[r-1]** i koristimo standardan algoritam na **a[l+1] .. a[r-2]**. Na ovaj način se opet izbegava najgori slučaj – konšćenje ograničavača (pominjali smo taj problem) jer biramo najzgodniji elemenat, a algoritam se dodatno ubrzava za oko 5%.

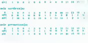
SORTIRANJE SA VELIKIM ZAPISIMA

Već smo primetili da sortiranje datoteka sa velikim zapisima uobičajenim metodama povlači veliki gubitak vremena na pomeranju velike količine podataka. Naravno, sada ćemo pokazati da je moguće sortirati bilo kojom metodom datoteke sa zapisima proizvoljne dužine sa samo **n** pomeranja zapisa!

Ideja se svodi na sledeće: umesto da pomera zapise, algoritam će pomerati pokazivače, a tek po završetku rada ćemo i zapise postaviti na svoje mesto. Ako niz **a[1] .. a[n]** sadrži **n** velikih zapisa, formiraćemo niz **p[1] .. p[l]** koji je definisan sa **p[i] = i**. Algoritam (dakle bilo koji) će elementima sada pristupati sa **a[p[i]]** pri upoređivanju, odnosno sa **p[i]** pri pomeranju. Novi niz nazivamo indeksnim nizom, a sam algoritam, zapravo, sortira indekse ovog niza po odnosima zapise iz niza **a**.

Pogledajmo sada implementaciju *insertion* metode na listingu 9.

Očito, prvo smo oformili niz **p**, a pri upoređivanju smo koristili upravo objašnjeni štos. Na kraju ove procedure dobijamo niz **a** kakav je i bio, a niz **p** potpuno ispremeštan (što je ilustrovano na slici 4). Sada možemo, recimo, ispisivati elemente niza **a** opet preko indeks-niza, čime smo izbegli čak i onih **n** pomeranja zapisa, ili možemo na osnovu indeksa postaviti zapise na svoje mesto, odnosno konačno sortirati niz **a**. Dobra ideja koja odavde sledi je da bi bilo dobro sva



Listing 9a

```
premesti (int a[], int p[], int n)
  {
    int i,j,k temp;
    for (i=1; i<=n; i++)
      if (p[i]!=i)
        {
          temp=a[i]; k=i;
          do
            {
              j=k; a[j]=a[p[j]];
              k=p[j]; p[j]=j;
            }
          while (k!=i);
          a[j]=temp;
        }
  }
```

Listing 10

```
for (i=0; i<m; i++) count[i]=0;
for (i=1; i<=n; i++) count[a[i]]++;
for (i=1; i<m; i++)
    count[i]=count[i-1]+count[i];
for (i=n; i>=1; i--)
    b[count[a[i]]--]=a[i];
for (i=1; i<=n; i++) a[i]=b[i];
```

Sortiranje niza koji sadrži po nekoliko istih elemenata

procedure koje operišu sa nizom **a** prepraviti da to čine preko niza **p**.

Međutim, šta ako, pri premeštanju zapisa na njihovo stvarno mesto posle sortiranja, ne želimo da potrošimo ni bajt memorije više nego što niz **a** zauzima? Kako nemamo memorije za kopiju niza, moramo smisliti neki drugi način. Pogledajmo naš primer pažljivije (slika 4). Na prvo mesto u nizu zapisa dolazi elemenat niza **a** sa indeksom 7, odnosno **a[7]**. Sačuvamo li stari zapis sa prvog mesta, recimo u neku promenljivu **temp** i izvršimo pomeranje, ostaje nam rupa u nizu **a** (na mestu **p[1]**, odnosno 7). Na to sedmo mesto treba da dođe elemenat sa indeksom **p[p[1]]** (što je **a[8]**). Na osmo mesto onda dolazi **a[4]** i tako dalje. Sled razmišljanja se vidi iz primera:

temp=**a[1]**;
a[1]=**a[7]**;   **a[7]**=**a[8]**;
a[8]=**a[4]**;   **a[4]**=**a[9]**;
a[9]=**a[14]**;   **a[14]**=**a[11]**;
a[11]=**temp**

Ovim smo sedam elemenata postavili na svoja mesta. Ako ovakav proces nastavimo za sledeći zapis koji nije na svom mestu (**a[2]** kod nas), konačno ćemo dobiti sortiran niz "a" (elemente koje smo prebacili na konačno mesto označimo stavljajući **p[i]=i** – listing 9a).

Jasno, ovu metodu treba koristiti onda kada nam je bitan memorijski prostor, a ne neki delić vremena. Za male nizove bi bila mala potrošnja memorije za kopiju, pa je nerealno mučiti se sa gornjom metodom. Kod velikog niza sa velikim zapisima, kopija bi zauzela dobro parče prostora, što ovim štedimo.

...I SPECIJALAN SLUČAJ

Jedna interesantna situacija koja se može desiti je sortiranje niza koji sadrži po nekoliko istih elemenata ("sortiraj niz od n zapisa čiji su ključevi celi brojevi od nula do **m-1**"). Ideja je da se u prvom prolazu prebroje sva pojavljivanja elemenata, a u drugom na osnovu njih poređaju elementi u konačni raspored. Ovde ćemo koristiti niz **a** sa zapisima, niz **b** sa pokazivačima (ovaj niz se inicijalizuje sa **b[i]=i**) kao što je pokazano na listingu 10.

U prvoj **for** petlji inicijalizujemo **count** niz sa **m** mesta, druga petlja prebrojava ista elementa pa imamo **count[1]=6**, **count[2]=4**, **count[3]=1** i **count[4]=4**. Treća petlja izračunava adrese za sortiranje – **count[1]=6**, **count[2]=10**, **count[3]=11** i **count[4]=15**, konačno, četvrta petlja postavlja elemente gde pripadaju (recimo prema gornjem: dvojke su od 7 do 10, trojka od 11 do 11, četvorke od 12 do 14).

Ovaj metod radi izvanredno za datoteke ovog tipa (red složenosti možete sami odrediti "od oka"). Šta više, mi ćemo kasnije proširiti ovaj algoritam u jedan daleko bolji – Radix sort.

SKRIVANJE NEVIDLJIVOG

Serija tekatova o programiranju u CAD-u nastavlja se po planu: ovog meseca prelazimo na poligone, jednačine prave i ravni, skrivanje zaklonjenih linija i senčenje tela, a tu su i još dva nova modula aplikacije „Računari_CAD".

Miroslav Olenjin

Poligon gradimo iz skupa duži, a duž iz početne i krajnje tačke, da li time rasipamo memoriju računara s obzirom da se uvek krajnja tačka prethodne i početna tačka naredne ivice poligona podudaraju? I da i ne. Naime, u programima koji poligon definišu direktno skupom tačaka, a veze među njima njihovim redosledom, javlja se teškoća kod predstavljanja torodnih (prstenastih) poligona. Definisanje poligona posredno preko skupa duži nam omogućava njegovu *nekontinualnost*, a to je zajedno sa većom brzinom pristupa i olakšanim manipulisanjem dovoljna dobit za cenu povećanog utroška memorije.

BAZIČNA OPERACIJA

Bazična operacija u radu sa poligonima jeste utvrđivanje položaja zadate tačke u odnosu na njega: da li se nalazi unutar konture, na ivici ili izvan nje. Koristi se za tu svrhu opšta topološka teorema prema kojoj svaki n-dimenzionalni prostor polovi n+1 dimenzionalni prostor. Tačka deli pravu na dve poluprave, prava deli ravan, ravan deli prostor, 3D prostor razdeljuje 4D prostor na dve dele itd. Da bi se prešlo iz jedne polovine hiperprostora u drugu, mora se preseći hiperravan neparan broj puta. Kod slučaja poligona u ravni položaj neke tačke u odnosu na njega ljudsko oko određuje „samo od sebe", intuitivno, al kompjuter nasuprot tome mora strpljivo da prebrojava stranice poligona koje se seku sa duži povučenom od zadate tačke do neke druge koja je sigurno izvan njega. Na slici 1 imamo prikazanu jednu situaciju kada se naše oko zbog konfiguracije konture dvoumi oko položaja tačke M u odnosu na poligon P, kompjuter međutim rutinski broji sedam preseka duži M-S sa njegovim stranicama i zaključuje da je M unutra. Štaviše, princip je toliko dobar da važi ne samo za konveksne i nekonveksne poligone već i za one koji su sačinjeni iz dve ili više odvojenih celina.

Za određivanje jednačine ravni u prostoru kojoj pripada poligon potrebne su nam i dovoljne tri njegove tačke. Da bi se došlo do poznatog opšteg oblika jednačine

$$Ax + By + Cz + D = 0 \tag{10}$$

moramo da izračunamo koeficijente A, B i C i slobodni član D. Činimo to razvijanjem sledeće determinante trećeg reda

$$\begin{vmatrix} X-x1 & Y-y1 & Z-z1 \\ x2-x1 & y2-y1 & z2-z1 \\ x3-x1 & y3-y1 & z3-z1 \end{vmatrix} = 0 \tag{11}$$

Brojevi A, B i C daće koordinate vektora normale na ravan, a preko D nalazimo njenu udaljenost od koordinatnog početka. Razmotrimo i uslove za kolinearnost i komplanarnost zadatih skupova tačaka u prostoru. Da bi tri tačke bile kolinearne, znači da bi ležale na istoj pravoj, determinanta matrice sačinjene od kombinacije njihovih razlika mora biti nula

$$\begin{vmatrix} x1-x2 & y1-y2 & z1-z2 \\ x2-x1 & y2-y3 & z2-z3 \\ x3-x1 & y3-y1 & z3-z1 \end{vmatrix} = 0 \tag{12}$$

Proširimo li ovu matricu novom tačkom, dobijamo analogan izraz za uslov komplanarnosti četiri tačke u prostoru. Dakle, kvartet tačaka leži u istoj ravni ako je zadovoljeno

$$\begin{vmatrix} 1 & x1-x2 & y1-y2 & z1-z2 \\ 1 & x2-x3 & y2-y3 & z2-z3 \\ 1 & x3-x4 & y3-y4 & z3-z4 \\ 1 & x4-x1 & y4-y1 & z4-z1 \end{vmatrix} = 0 \tag{13}$$

Sa jednačinom prave ćemo najlakše baratati u njenom kanonskom obliku jer se određuje na osnovu dve njene tačke u prostoru.

$$\frac{X-x1}{x2-x1} = \frac{Y-y1}{y2-y1} = \frac{Z-z1}{z2-z1} \tag{14}$$

Delioci iz ova tri razlomka označavaju se skraćeno sa p, q i s respektivno. Nadalje, izrazi za prodor prave u ravan se izvode rešavanjem sistema jedne ravni i jednog svežnja ravni, a konačne formule glase ovako

$$X = x1 - \frac{Ax1 + By1 + Cz1 + D}{Ap + Bq + Cs} * p \tag{15}$$

$$Y = y1 - \frac{Ax1 + By1 + Cz1 + D}{Ap + Bq + Cs} * q \tag{16}$$

$$Z = z1 - \frac{Ax1 + By1 + Cz1 + D}{Ap + Bq + Cs} * s \tag{17}$$

Srednji razlomak je u sva tri izraza isti, tako da se uvek prvo on izračunava. Prikažimo još i jednačinu za normalno odstojanje tačke od ravni

$$d = \frac{Ax + By + Cz + D}{\pm\sqrt{(A^2 + B^2 + C^2)}} \tag{18}$$

Kao i u slučaju slične ravanske formule koju smo dali u prošlom broju, i ovde predznak rezultata zavisi od toga da li data tačka i koordinatni početak leže na istoj (d < 0) ili suprotnim (d > 0) stranama ravni. Što se tiče izraza za računanje uglova između dve ravni, dve prave ili prave i ravni, na njima se nećemo zadržavati jer se svi svode na nalaženje ugla između para vektora, do čega se lako dolazi iz njihovog skalarnog proizvoda.

HIDE RUTINA

Eliminisanje zaklonjenih linija najelementarnija je najranija tehnika realističnog prikaza tela u prostoru. U velikoj meri je uslovljena strukturom podataka kojima se determiniše telo: obični žičani modeli ne zadovoljavaju nego je potrebno da rutina za eliminaciju zaklonjenih ivica (engl. *removal hiding edges routine* ili *HIDE routine*, prema nazivu odgovarajuće opcije u CAD paketima) ima na raspolaganju informacije o poligonima kojima se i vektori objedinjuju. Sve aktuelne strukture memorisanja definicija tela (Boundary, CSG) spunjavaju ovaj preduslov, ali se sami algoritmi sa svojim performansama i pouzdanošću drastično razlikuju baš u zavisnosti od njih. Autori prvih verzija CAD paketa za personalne računare vrlo često su se odlučivali da kod svojih rutina žrtvuju malo pouzdanosti zarad brzine – i sam *AutoCAD* je sve do skora znao da „progute" ili „izmisli" poneku linijicu na objektu koga obrađuje.

Mi ćemo ovde ukratko izložiti klasični algoritam kojeg odlikuju tačnost i univerzalnost – radi i sa konveksnim i nekonveksnim poligonima proizvoljne složenosti – kao i primerenost strukturi podataka u aplikaciji Računari_CAD. Algoritam je kvadratni jer se zasniva na poređenju na ukrštanje svake duži svakog poligona sa svim drugim dužima koje ne pripadaju tom poligonu, vreme izvršavanja međutim veoma zavisi ne samo od broja duži već i od njihovog uzajamnog položaja u projekcionoj ravni. Verzija aplikacije čiji kompletan listing donosimo u ovoj seriji članaka ne sadrži eliminisanje zaklonjenih ivica, kao uostalom ni sijaset drugih standardnih opcija. I ovakva kakva je, sa svojih 900 linija u *Turbo Pascalu V6* predstavlja ubedljivo najduži program objavljen u našem časopisu. Ipak, ona je jedan čvrst i stabilan skelet oko koga možete eksperimentisati i nadovezivati sopstvene zamisli, a zahvaljujući objektnoj organizaciji aplikacije to bi trebalo ići lako i očigledno – barem što se *HIDE* rutine tiče. Najbitnije je kod svake od klasa razlučiti šta se od primljenih naredbi „odozgo" odmah izvršava u okviru te klase, a šta se samo prenosi kao informacija skupinama nižih, podređenih klasa.

Razmotrimo sliku 2. Na njoj su prikazane tri faze uklanjanja nevidljivih ivica poligona **abcdefgh** zaklonjenih poligonom **ABCDE**. Problem koga potencira položena ilustracija je pojava delimične zaklonjenosti, tj. segmentiranja svake duži po presecima sa drugima. *HIDE* rutini bivaju u jednoj petlji prosleđivane početne i krajnje tačke svake duži, uz prethodnu proveru nalazi li se ona bar delimično u vidnom polju, tj. prozoru na ekranu gde se usmerava iscrtavanje. Krećemo od prve duži prvog poligona **abcdefgh**, tj. a-b, koju treba logičkom funkcijom sličnom **Cross** rutini opisanoj u prethodnom nastavku serije uporediti (da li se seče) redom sa svim ivicama poligona **ABCDE**. Odmah uočavamo korisnu heuristiku: ako ispitivana duž nikako ne dotiče ili ne prolazi kroz pravougaono područje kojim je omeđen poligon, onda nema smisla ni da se seče sa bilo kojom njegovom ivicom. U slučaju duži a-b možemo zanemariti postojanje cele **ABCDE** konture i direktno ići na poređenje sa sledećim poligonima, ili ako ih nema, odmah iscrtati a-b u punoj veličini. Već naredna duž b-c ne ispunjava prethodni uslov, pa je moramo porediti sa svakom duži drugog poligona. Da li se seče sa A-B? Ne seče se. Da li se seče sa B-C? Da, seče se – sada treba rešavanjem sistema dve linearne jednačine sa dve nepoznate naći presečnu tačku, podeliti b-c na dva segmenta, na b-6 i na 6-c, i za njih dalje nastaviti postupak poređenja sa svakim segmentom zasebno. Jasno je da ovde do izražaja dolazi rekurzija: *HIDE* rutina vrši dva autopoziva (grad lako prvi čvor binarnog stabla) sa parametrima početnih i krajnjih tačaka dve nove duži. Novci nicirane rutine za svaku od njih dalje nastavljaju njihovo upoređivanje sa preostalim dužima i poligonima – konkretno, za segment b-6 se ustanovilo da se ne seče ni sa jednom drugom duži, što znači da je ili potpuno vidljiv ili potpuno zaklonjen.

Sad dolazimo do dileme da li ga iscrtati ili ne. Uzima se bilo koja tačka na dotičnom segmentu, najzgodnije ona na sredini, i redom ispituje da li ona u prikazu na ravni perspektivne projekcije leži u nekom od poligona. Kada se ustanovi da jeste unutar nekog poligona, prelazi se na proračun sa 3D koordinatama: nalazi se presečna tačka prave koja povezuje gledište

Slika 1



Optičke varke: Gde se nalazi tačka M u odnosu na poligon?

Listing 1 Klasa TPoly

```
{
 +---------------------------------------------------+
 |            Aplikacija 'RacunariCAD'                |
 |   Klasa TPoly: definiše objekat poligona u prostoru |
 |        Miroslav Olenjin  (c) mart/Avgust 1992      |
 +---------------------------------------------------+
}
unit  CPoly;    {$A+,F+,O-,B-,S-,R-,IFDEF CPU87 N+} {$ENDIF}

INTERFACE

uses  CLine;

type  TPolyPtr = ^TPoly;
      TPoly = object
        As, Bs, Cs, Ds, Dist, Defl  : real;
        Xmin, Xmax, Ymin, Ymax      : integer;
        FirstLine                   : TLinePtr;
        Next                        : pointer;
        constructor Init;
        function Min(r1,r2:real):integer;
        function Max(r1,r2:real):integer;
        procedure View(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz,Dx:real);
        function Intersection(Px,Py:integer):Boolean;           virtual;
        procedure Show(OriginX,OriginY:integer;Cf:real);
        procedure Move(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz:real);
        procedure AppLine(xt,yt,z1,x2,y2,z2:real);
        destructor Done;
      end;

IMPLEMENTATION

var  Persp, x1, y1, z1, x2, y2, z2, x3, y3, z3 : real;
     LineCount : TLinePtr;
     CutCount  : word;

constructor TPoly.Init;
begin  FirstLine:=nil;  end;

function TPoly.Min;
begin  if r1 < r2 then Min:=Round(r1) else Min:=Round(r2);  end;

function TPoly.Max;
begin  if r1 > r2 then Max:=Round(r1) else Max:=Round(r2);  end;

procedure TPoly.View;
begin
  Persp:=Dx; LineCount:=FirstLine;
  LineCount^.View(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz,Dx);
  Xmin:=Round(LineCount^.p1.Rx); Xmax:=Xmin;
  Ymin:=Round(LineCount^.p1.Ry); Ymax:=Ymin;
  while LineCount <> nil do begin
   with LineCount^ do begin
     View(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz,Dx);
     Xmin:=Min(Min(p1.Rx,p2.Rx),Xmin);
     Xmax:=Max(Max(p1.Rx,p2.Rx),Xmax);
     Ymin:=Min(Min(p1.Ry,p2.Ry),Ymin);
     Ymax:=Max(Max(p1.Ry,p2.Ry),Ymax);
   end;
    LineCount:=LineCount^.NextLine;
  end;
  LineCount:=FirstLine;
  with LineCount^ do begin x1:=p1.Rx; y1:=p1.Ry; z1:=p1.Rz; end;
  LineCount:=LineCount^.NextLine;
  with LineCount^ do begin x2:=p1.Rx; y2:=p1.Ry; z2:=p1.Rz; end;
  LineCount:=LineCount^.NextLine;
  with LineCount^ do begin x3:=p1.Rx; y3:=p1.Ry; z3:=p1.Rz; end;
  As:=((y2-y1)*(z3-z1))-((y3-y1)*(z2-z1));
  Bs:=((x3-x1)*(z2-z1))-((x2-x1)*(z3-z1));
  Cs:=((x2-x1)*(y3-y1))-((x3-x1)*(y2-y1));
  Ds:=-As*x1-Bs*y1-Cs*z1;
  Defl:=Abs(As/Sqrt(Sqr(As)+Sqr(Bs)+Sqr(Cs)));
end;

function TPoly.Intersection;
begin
  Intersection:=false;
  if (Px>=Xmin) and (Px<=Xmax) and (Py>=Ymin) and (Py<=Ymax)
  then begin
    CutCount:=0;
    LineCount:=FirstLine;
    while LineCount <> nil do begin
      if LineCount^.Cross(Px,Py,Xmax+1,Ymax+1) then inc(CutCount);
      LineCount:=LineCount^.NextLine;
    end;
    if Odd(CutCount) then begin
      Dist:=Persp+((As*Sqr(Persp)+Ds*Persp)/(As*Persp+Bs*Px+Cs*Py));
      if Dist <= 0 then Intersection:=true;
    end;
  end;
end;

procedure TPoly.Show;
begin
  LineCount:=FirstLine;
  while LineCount <> nil do begin
    LineCount^.Show(OriginX,OriginY,Cf);
    LineCount:=LineCount^.NextLine;
  end;
end;

procedure TPoly.Move;
begin
  LineCount:=FirstLine;
  while LineCount <> nil do begin
    LineCount^.Move(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz);
    LineCount:=LineCount^.NextLine
  end;
end;

procedure TPoly.AppLine;
begin
  New(LineCount,Init(x1,y1,z1,x2,y2,z2));
  LineCount^.NextLine:=FirstLine;
  FirstLine:=LineCount;
end;

destructor TPoly.Done;
begin
   while FirstLine <> nil do begin
     LineCount:=FirstLine^.NextLine;
     Dispose(FirstLine);
     FirstLine:=LineCount;
   end;
 end;

 end.
```

Klasa definiše osnovu objekta tela u prostoru

se središnjom tačkom ispitivanog segmenta i ravni kojoj pripada poligon. Ako se nađe da je ta presečna tačka bliža posmatraču od tačke na segmentu, onda se smatra da je on zaklonjen i ne iscrtava se. U suprotnom, ponavlja se isto ispitivanje za iduće poligone sve dok se ne iscrpi njihova lista. Ako se prošlo kroz celu listu poligona, znači da segment nije ničim zaklonjen pa ga shodno tome možemo prikazati povlačenjem linije.

Primetimo da ovakav algoritam neće korektno raditi kada se u prostoru desi uzajamno presecanje dva poligona. Mora se uneti i provera za svaku ispitivanu duž, odnosno segment duži, da li tačka preseka njene prave nosača i ravni poligona pripada domenu duži. Ako pripada, znači da ta duž probija poligon, te da ne može biti u celosti vidljiva, pa je treba podeliti na dva segmenta, i to pre nego se počne poređenje sa ivicama datičnog poligona.

Da bismo ovo konkretizovali i dopunili, pozvaćemo se na sliku 3. Tu se vide dva uzajamno presecajuća nekonveksna poligona u obliku slova „c", prikazana opet kroz neke in faze eliminisanje zaklonjenih ivica. Sa aspekta dosadašnjeg izlaganja, faze broj dva je konačna – međutim uočljivo je da ona, iako smisaono ispravna, nije nekako vizuelno najidealnija. Zapazite: treba posebnim linijama kojih inače nema definisanih u bazi ivica, međusobno spojiti tačke preseka među poligonima kako bi se dočaralo da su posredi pune površine a ne ramovi od žice. E sad, problem je u redosledu povezivanja tih tačaka (kojih je uvek paran broj, a ilustracija sugeriše da ih može biti i više od dve). Recimo, imamo kao na slici četiri presečne tačke, povezuju se 1 i 2 i povezuju se 3 i 4. Ali kako će HIDE rutina znati koja je prva, koja druga, treća, četvrta? Ili, u opštem slučaju, n-ta? Jer rutina do njih ne mora dolaziti onim redosledom kako one geometrijski stoje na preseku ravni poligona.

Očigledno je da su sve te tačke kolinearne, da jedna za drugom leže na istoj pravoj i da je reč o njihovom sortiranju u takav poredak da ih kasnije možemo povezivati dužima, svaku neparnu sa sledećom u nizu. Međutim, nije baš lako dokučiti po kom kriterijumu vršiti sortiranje — ako uvedemo, na primer, da nam je kriterijum za uređivanje dobijenog skupa tačaka njihova x koordinata, to neće funkcionisati za prave normalne na x osu. Analogno važi i ako kao kriterijum uzmemo y ili z koordinatu. Jasno je da moramo uzeti u obzir sve tri koordinate, ali kako? Kao zbir x+y+z? Ni to neće biti dobro – primence tačke (1 1 1), (2 2 2) i (3 3 3) su kolinearne i već sortirane prema položaju na svojoj pravoj, ali im zbir koordinata to ne govori. Jeste da se slučajevi kada je pravašaj zbir koordinata jednak nuli javljaju samo u specifičnim položajima prave nosača, no ipak bi bilo pogrešno zanemariti ih i zato se uvode drugačiji pristupi. Jedan od poznatijih predviđa da se kroz zadatu tačku postavi jedna pomoćna ravan, normalna na pravu nosača, rastojanje te ravni od neke fiksirane tačke, najbolje od koordinatnog početka, služi u procesu sortiranje kao pouzdan kriterijum.

Rešimo još jedan problem u vezi sa eliminisanjem zaklonjenih ivica. U CAD razvojnim paketima opremljenim HIDE rutinom obično postoji mogućnost redirekcije rezultata njenog izvršavanja sa ekrana u datoteku. Sa tom datotekom koja sadrži vektorski opis pogleda objekta sa uklonjenim nevidljivim ivicama, rad na crtežu se najčešće nastavlja u nekom *2D CAD* programu (*DC3 —> DC2* ili *AutoCAD —> AutoSketch*) da bi mu se tamo dodali okvir, sastevnice, legenda, impresum, izvršilo šrafiranje, kotiranje, dimenzija itd. Naposletku bi se vektorski opis crteža slao ploteru. Elem, nedorađena *HIDE* rutina zna sve ove poslove da zakomplikuje, uspon time što napravi izvoznu datoteku nekoliko puta veću od datoteke sa prvobitnim prikazom žičanog modela, iako bi se zdravorazumski pomislilo da više vektora sadrži ova druga! Da bi razumeli zašto se to dešava, pogledajmo opet sliku 2. Pri segmentiranju duži B C one bive izdeljena na šest delića i njedan nije zaklonjen. Svi se dakle iscrtavaju, odnosno po zahtevu upisuju u datoteku, mada bi se mogli zameniti jednom jedinom duži početnom B C. Kako sad u okviru *HIDE* rutine prepoznati nepotrebno segmentiranje, odnosno ubrzati iscrtavanje i izbeći da se na disk šalje višak vektora i da glava plotera istu liniju vuče iz nekoliko poteza?

Treba je u osnovi napisati kao logičku funkciju *Tačno* (True) bi vraćale onde kada utvdi da je segment duži koji joj je prosleđen u potpunosti vidljiv, a to bi bilo u dva slučaja: prvo, ako nije došlo do razdeljivanja segmenta, onda je on vidljiv ukoliko nije zaklo-

Listing 2 Klasa TBody

```
{
 ------------------------------------------------------------
|                 Aplikacija "RenneriCAD"                    |
|   Klasa TBody  definise osnovn objekte tela u prostoru.    |
|         Miroslav Olenjin  (c) Mart/Avgust 1992.            |
 ------------------------------------------------------------
}
unit  CBody;    {$A+,F+,B-,R-,S-,V-,IFDEF CPU87 N+} {$ENDIF}

INTERFACE

uses  Cpoly;

type  TBodyPtr = ^TBody;
      TBody = object(TPoly)
        Ident        : string[10];
        Exten        : string[4];
        FirstPoly    : TPolyPtr;
        procedure MinMax(Count:TPolyPtr);
        function CmpDistance
             (Inter:Boolean;Count:TPolyPtr;Px,Py:integer):Boolean;
        procedure View(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz,Ox:real); virtual;
        procedure Show(OriginX,OriginY:integer;Cf:real);    virtual;
        function Intersection(Px,Py:integer):Boolean;        virtual;
        procedure Move(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz:real);    virtual;
        procedure Triangle(x1,y1,z1,x2,y2,z2,x3,y3,z3:real);
        procedure Square(x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4:real);
        procedure List(Tab:string);
        destructor Done;
      end;


IMPLEMENTATION

var  PolyCount,NewPoly : TPolyPtr;

procedure TBody.MinMax;
begin
  Xmin:=Min(Count^.Xmin,Xmin);  Xmax:=Max(Count^.Xmax,Xmax);
  Ymin:=Min(Count^.Ymin,Ymin);  Ymax:=Max(Count^.Ymax,Ymax);
end;

function TBody.CmpDistance;
begin
  if (Px>=Xmin) and (Px<=Xmax) and (Py>=Ymin) and (Py<=Ymax) then
    if Count <> nil then begin
      while Count <> nil do begin
        if Count^.Intersection(Px,Py) then begin
          if not(Inter and (Dist > Count^.Dist)) then begin
            Dist:=Count^.Dist;
            Refl:=Count^.Refl;
          end;
          Inter:=true;
        end;
        Count:=Count^.Next;
      end;
    end;
  CmpDistance:=Inter;
end;

procedure TBody.View;
begin
  FirstPoly^.View(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz,Ox);
  Xmin:=Round(FirstPoly^.Xmin); Xmax:=Xmin;
  Ymin:=Round(FirstPoly^.Ymin); Ymax:=Ymin;
  PolyCount:=FirstPoly;
  while PolyCount <> nil do begin
    PolyCount^.View(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz,Ox);
    MinMax(PolyCount);
    PolyCount:=PolyCount^.Next;
  end;
end;

procedure TBody.Show;
begin
  PolyCount:=FirstPoly;
  while PolyCount <> nil do begin
    PolyCount^.Show(OriginX,OriginY,Cf);
    PolyCount:=PolyCount^.Next;
  end;
end;

function TBody.Intersection;
begin
  Intersection:=CmpDistance(False,FirstPoly,Px,Py);
end;

procedure TBody.Move;
begin
  PolyCount:=FirstPoly;
  while PolyCount <> nil do begin
    PolyCount^.Move(Sa,Sb,Sg,Ca,Cb,Cg,Tx,Ty,Tz);
    PolyCount:=PolyCount^.Next;
  end
end

procedure TBody.Triangle;
begin
  New(PolyCount,Init);
  PolyCount^.Next:=FirstPoly;
  FirstPoly:=PolyCount;
  with FirstPoly^ do begin
    AppLine(x1,y1,z1,x2,y2,z2);
    AppLine(x2,y2,z2,x3,y3,z3);
    AppLine(x3,y3,z3,x1,y1,z1);
  end;
end;

procedure TBody.Square;
begin
  New(PolyCount,Init);
  PolyCount^.Next:=FirstPoly;
  FirstPoly:=PolyCount;
  with FirstPoly^ do begin
    AppLine(x1,y1,z1,x2,y2,z2); AppLine(x2,y2,z2,x3,y3,z3);
    AppLine(x3,y3,z3,x4,y4,z4); AppLine(x4,y4,z4,x1,y1,z1);
  end;
end;

procedure TBody.List;
begin
  WriteLn(Tab,Ident:10,'  ',Exten:4);
end;

destructor TBody.Done;
begin
  while FirstPoly <> nil do begin
    PolyCount:=FirstPoly^.Next;
    Dispose(FirstPoly,Done);
    FirstPoly:=PolyCount;
  end;
end;

end.
```

Klasa definise objekat poligone u prostoru

njen ni jednim poligonom drugo, ako jeste došlo do segmentiranja onda valja sačekati da obe rekurzivno pozvane *HIDE* funkcije vrate *tačno* da bi i ona vratila *tačno*, u suprotnom se vrši iscrtavanje ne akcan ili dodavanje novog sloga u datoteku za onaj segment duž za koji je povraćena vrednost *tačno*. Primetimo da kad neka od dve rekurzivno pozvane funkcije vrati *netačno (False)*, njoj prosleđeni segment se ne iscrtava. Jako ne mora da znači da je u potpunosti zaklonjen. Moguće je da ga je ona dalje delila i da su neki od njih još manjih delića vidljivi a neki ne – no svejedno posao oko iscrtavanja je tada već bio obavljen.

SHADE RUTINA

Pod terminom senčenja prostornih objekata podrazumevamo veoma mnogo tehnika prikaza ispunjenih površina – pomenimo samo **FlatShade**, **Phong**, **Gouraud**, **Brilliance**, različitih po složenosti, kvalitetu i brzini. Možemo ih po načinu rada svrstati u dve grupe.

Prva grupa tehnika bazira se na algoritmu suštinski sličnom *HIDE* rutini, koji man puise nad poligonom kao geometrijskom celinom tako što svaki od poligona tela poredi sa svim ostalima u smislu da li ih potpuno ili delimično preklapa. Ako je ovo drugo, ispitivani po-



ligon se „kroji" na potreban broj delova po liniji drugog poligona, pa se postupak dalje nastavlja za svaki deo pojedinačno (opet rekurzivno) sve dok se ne dođe do kraja liste, odnosno delića koji su ili potpuno zaklonjeni ili potpuno vidljivi. Konačno se za svaki od njih ispituje postoji li neki drugi poligon u tom pravcu isprečan između njega i posmatrača ta se na osnovu tog odgovora on iscrtava ili ne iscrtava. Iscrtava se naravno ispunjena kontura u pripadajućoj boji ili šrafuri.

Mada je i ovaj opisani algoritam kvadratno nepogodan za obradu većeg broja poligona, pak je veoma efikasan u slučajevima prostijih tela. U paketima za projektovanje novijeg datuma može se sresti kao opcija za brzu proveru položaja objekata pred startovanje kvalitetnog senčenja. Jedna druga optimizovana izvedba ovog algoritma ograničena na jednostavne nekonveksne poligone i sa njihovim unapred izraču-

nalim položajima, nazvana je **FREESCAPE** i primenjena u video igrama sa takozvanom „filovanom vektorskom grafikom" za dočaravanje kretanja u tri dimenzije, mnoge od takvih igara kao što su *Driller*, *Star Glider* ili *Flight Simulator* postigle su velike tiraže.

PIKSEL PO PIKSEL

Sasvim drugačije rastersko (bit mapirano) senčenje podrazumeva prvo redukovanje projekcione ravni na pravougaonik veličine ekrana, odnosno prozora u kome se vrši iscrtavanje. U dvostrukoj petlji (sleva na desno, red za redom) određuju se boja i osvetljenost svakog piksela tako što se gleda da li zrak koji povezuje tačku gledišta i projekcionu ravan na poziciji tog piksela „pogađa" neki od poligona tela. Ako zrak prolazi kroz više njih, meri se udaljenosti do presečnih tačaka i pikselu na ekranu dodeljuje boja najbližeg poligona, a u intenzitetu zavisnom od upadnog ugla. Ako izvor svetlosti nije slučajno podudaran sa tačkom posmatranja, valja još jednom proći kroz listu poligona i ispitati da li neki drugi stoji isprečen između izvora svetlosti i presečne tačke i time „baca senku" na nju, pa na osnovu toga korigovati jačinu boje piksela. Prirodno, ako imamo ne jedan nego nekoliko izvora sve-



tosti proizvoljno razmeštenih u prostoru proceduru treba ponavljati za svaki od njih a konačan rezultat izvesti aritmetičkom sredinom (Metoda aritmetičke sredine se kod prisustva opcije za slobodno podešavanje boje svetlosti iz izvora ograničava samo na intenzitet, dok se kolorit novodobijene boje izvodi tako zvanim RGB računom, npr. žuta + plava = zelena.) Bolji CAD-ovi poseduju i opciju definisanje stepena transparentnosti tela, kada se ispitivanom pikselu na dodeljuje vrednost na osnovu svojstava samo najbližeg poligona već i drugih koji se „provide" kroz njega.

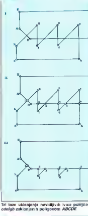
Ovakav algoritam je (više manje) linearan u funkciji broja poligona i rezolucije scene, daje lepe i tačne prikaze, ali nikada nije preterano brz. To i ne treba da čudi s obzirom na ogroman broj repeticija poređenja udaljenosti poligona: pri rezoluciji od 1024x768 (a to je tak minimum za isplativ rad) kroz ceo gore opisan postupak treba proći više od sedamsto pedeset hiljada puta! Sa druge strane, pojedinačno nijedna od ovoliko sličnih operacija ne zavisi od prethodnih ni narednih, tako da je rastersko senčenje idealno za brzo paralelno procesiranje: na profesionalnim radnim stanicama sve češće se viđaju posebna elektronska kola za hardversko senčenje, koja uz pomoć pajplajnovanja, baferovanja, keširanja i sličnih smicalica postižu da se osenčana slika visoke rezolucije generiše maltene trenutno!

IZVAN NAŠEG DOMAŠAJA

Ray-trace ili metoda senčenja putem praćenja trajektorije svetlosnih zraka, daje najrealističnije scene, ali i zahteva enormne računarske resurse. Aplikacije koje podržavaju rejtrejsing senčenje dopuštaju da se svakom telu zadaju njegove optičke osobine: koliko procenata od intenziteta upadnog zraka će površ da reflektuje, koliko da apsorbuje, a koliko će propustiti dalje. Najbolji kvalitet se postiže kada se tela opisuju ne aproksimacijom sa velikim brojem malih poligona već matematičkim jednačinama površi – tako se i na zaobljenjima dobijaju savršeni odrazi. Odgovarajući algoritam pri nailasku na telo svaki zrak rastavlja na određen broj novih, rukovodeći se pripremljenim matematskim formulama. Uprošćeno govoreći, zrak koji je odbijen od predmeta biće pod istim uglom u odnosu na vektor gradijent funkcije površi kao i upadni zrak i u istoj ravni sa njima, a zrak koji prodire unutar tela skrenuće sa putanje prema Dekart-Snelijusovom zakonu, što će reći u zavisnosti od količnika gustina dva sredina. Svakom novonastalom zraku računar mora dalje da izračunava put, sve dotle dok ovaj ne „naleti" na 100% apsorbujuću površ ili beskonačan prazan prostor, odnosno dok mu intenzitet tokom prelamanje ne oslabi ispod razlike među nijansama boja korišćene grafičke kartice.

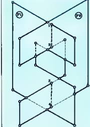


Slika 2



Tri faze uklanjanja nevidljivih ivica poligona *abcdefgh* zaklonjenih poligonom *ABCDE*

Slika 3



Dva nekonveksna poligona u obliku slova „c" koji se uzajamno preklapaju prikazana kroz tri faze eliminisanja zaklonjenih ivica



Najnoviji high-end softver (sa cenama od 100 000 dolara naviše) prevazilazi čak i granicu između geometrijske i fizičke optike i tretira svetlost kao talase, što ona stvarno i jeste. Takvim softverom se mogu verno simulirati pojave karakteristične za talasno kretanje svetlosti, poput refrakcije, disperzije i distorzije. Recimo, svetlost koja prođe kroz usku rešetku biće polarizovana, slamka u čaši limunade prikazana prelomljena, iz prozirne prizme šinće se dugine boje, a kroz naočari ostavljene na otvorenoj knjizi jasno će se videti uvećana slova!

KLASA TPoly

Trenutak je da se vratimo na ono „od čega se živi" – na listinzima nas čekaju dva nova modula, dve nova klase aplikacije „Računari-CAD". Prva od njih klasa **TPoly**, objedinjuje proizvoljan broj objekata klase **TLine**, tj. duži u prostoru. Pošto sve te duži leže u istoj ravni (komplanarne su), osnovni je zadatak klase **TPoly** nalaženje jednačina te ravni i na osnovu nje, u kasnijim operacijama, izračunavanje odstojanje poligona od tačke gledišta i ugla pod kojim ga vidi posmatrač.

Od polja u klasi **TPoly** imamo prvo varijable **Aa**, **Ba**, **Ca** i **Da** će sadržavati koeficijente jednačine ravni poligona, **Dist** promenljivu koristi metoda **Intersection** za smeštanje rezultata svog rada, varijabl **Refl** metoda **View** dodeljuje sinus ugla koji zaklapa ravan poligona sa x osom, **FirstLine** pokazuje na prvi objekat (glavu liste) klase **TLine** u listi duži koje čine poligon, pointer **Next** čini vezu sa narednim objektom **TPoly** u listi poligona.

Klasa **TPoly** za sebe i svoje naslednice uvodi i četiri indžer promenljive za smeštanje koordinata pravougaonog područja vidljivosti odgovarajućeg geometrijskog entiteta na projekcionoj ravni. To će reći da se u ovoj klasi one odnose na poligon, u klasi **TBody** na kompletno telo, a u klasi **TGroup** na celu skupinu tela i drugih grupa. Postoje dva razloga za uvođenje ovih granica vidljivosti: da bi u klasi **TPoly** pri ispitivanju položaja date tačke u odnosu na poligon lakše došli do neke koja je sigurno izvan poligona, i drugo, da bi ubrzali opcije koja imaju smisla samo za vidljiv deo crteža. Na primer, kod iscrtavanja ili senčenja, ako zaključimo da je neki poligon/telo/skupina sasvim izvan ekrana, nećemo se njima ni baviti.

Dakle, konkretno u ovoj klasi **Xmin** će odrađivanjem **View** metode primiti vrednost krajnje leve x koordinata poligona, **Xmax** krajnje desne, **Ymin** najgornja y koordinate, a **Ymax** najdonje. Njih će naročito intenzivno ispitivati metoda **Intersection**, pa su u cilju povećanja brzine manipulisanja deklarisane kao celobrojne. Kako su i koordinate ekrana celobrojne, poneke mala gubljenje na tačnosti pri zaokruživanju *real* vrednosti neće se pri procesu senčenja odraziti na tačnost slika.

Pored varijabli u pojma klase, jedinica **CPoly** inicira još nekoliko varijebl koja kao „opšte dobro" koriste svi primerci objekata tipa **TPoly**. Izdvojene su iz pod *Implementation* sekcije jer nema potrebe da se prave posebne kopije za svaki objekat, a i pristup im se ubrzava jer se zapisuju u DATA segment. Redom ih nabrajajući: **Persp** čuva udaljenost posmatrača za metodu **Intersection**, promenljiva x1, y1 ... z3 pomažu metodi **View** kod nalaženja jednačine ravni, **LineCount** se koristi za smeštanje privremenog pokazivača u sukcesivnim operacijama nad listom stranica poligona, a **CutCount** broji preseke pomoćne duži sa njima.

Metoda su sledeće. Konstruktor **Init** setuje pokazivač **FirstLine** na **nil** – lista će se tek kasnije popunjavati dužima, u funkcijama **Min** i **Max** su izdvojene

NASTAVLJA SE NA STRANI 84

BATCH INTERPRETER ZA WINDOWS

Broj *Windows* API funkcija je veoma veliki i može vam se desiti da nakon nekoliko desetina napisanih programa još uvek i ne dotaknete neke oblasti. U našim dosadašnjim prilozima nije bilo reči o jednog grupi veoma korisnih funkcija poznatih pod zajedničkim nazivom *MetaFile*.

Dejan Dinčić

MetaFile funkcije omogućavaju kreiranje fiktivnog *display context*-a (DC). Tako kreiran DC ima nekoliko važnih osobina:

- na njemu se mogu izvoditi sve **GDI** (grafičke) funkcije
- sva izvedena operacije nad ovim DC-om pamte se u *MetaFile*-u
- jedna ili niz operacija zapamćenih u *MetaFile*-u mogu se reprodukovati u bilo kom drugom DC-u

Praktično, to znači da nam je na raspolaganju mehanizam za snimanje niza grafičkih operacija i njihovu reprodukciju. Primenu ovoj tehnici uopšte nije teško naći.

Prisetimo se načas kako *Windows* program crta. Crtanje se odvija ili kao reakcija na neke opcije programa – glavni tok programa – ili kada je nešto van programa (kao što je prekrivanje nekim drugim prozorom...) povisina prozora delimično ili u celini obrisana. U slučaju brisanja celine ili dela prozora, potrebno je restaurirati narušeni sadržaj prozora. Standardne elemente prozora (okvir, meni i slično) odnavlja *Window manager*, ali radnu površinu programa mora program da ispuni ponovo sam. Kod nekih programa obnavljanje sadržaja prozora nije problem. Treba jednostavno voditi uvek računa o stanju programa i ponovo nacrtati nekoliko objekata. To je naročito slučaj kod programa koji operišu jasnim skupom objekata i jasnim **redosledom** njihovog prikazivanja. Postoje i programi sa drugačijom koncepcijom ispisa. Linijski ispis teksta, kao u komunikacionom programu, zatim ispis čiji redosled (a time možda i vidljivost pojedinih elemenata) zavisi od interakcije sa korisnikom, kao i mnoge druge situacije kod kojih bi interno predstavljanje izvršenog ispisa bilo komplikovano ili glomazno, prosto vapiju za mehanizmom koji bi omogućio da se sav izvršeni ispis bez mnogo dodatnog programerskog napora i da se može prosto reprodukovati, na primer pritiskom na "play" dugme nekog video rekordera.

MetaFile stvarno funkcioniše ovako jednostavno, a u to se možete uveriti kroz programski primer koji smo obradili ovaj put. Reč je o programu koji predstavlja mali *batch* interpreter u *Windows* okruženju. Verovatno je i vama pri radu sa *Windows*-ima često nedostajala mogućnost da na pravite manje procedure nalik na *batch* procedure pod DOS-om, a da za lakve stvari ne potežete nikakav kompajler ili neko drugi alat s kojim morate proći ceo put razvoja jednog programa. *WinBatch* program koji prikazujemo omogućava interpretaciju procedura zapisanih u običnim tekst datotekama. Sintaksa koju uvodimo krajnje je jednostavna, a mogućnosti interpretera su ipak dovoljne da se njima realizuju raznovrsne procedure.

PROGRAM WINBATCH

Kao osnova za program i sintaksu procedure poslužio je sjajan program *Menu* i tekst Zorana Živkovića "Ja bih bi menu" u Računarima broj 71. *Menu* je nastao da bi pružio više od DOS *batch* procedura i, osim izrade menija, kroz napredniji korisnički interfejs omogućio kreiranje raznorodnih funkcija. Ponovno prelistavanje ovog teksta pomoći će vam da lakše pratite listing programa koji sada prikazujemo.

Program *WinBatch* zadržao je osnovnu postavku programa *Menu*: učitavanje konfiguracione datoteke, interpretacija i kontrola toka procedure su pretrpeli samo minimalne promene, tako da ih nećemo detaljno analizirati ovom prilikom. Bitna izmena u samoj osnovi programa je napravljena u rešenju dinamičkog odvajanja memorije (primenjeno je rešenje saglasno *Windows* okruženju). Pored toga, uočava se bitna razlika u globalnoj strukturi programa jer je, narav-

no, ubačena petlja za poruke i *Window* procedura koja ih obrađuje.

Što se tiče sintakse procedura koje naš program interpretira i ona je pretrpela određene izmene. Evo koji su osnovni elementi "programa" koji se interpretira:

- kontrola toka (labele i naredbe goto, call i end)
- string promenljive
- interne naredbe
- startovanje eksternih programa

Startovanje eksternih programa izaziva malu razliku u koncepciji: dok je izvršavanje procedure pod DOS-om prekinuto za vreme izvršavanja spoljnog programa i nastavlja se na mestu prekida po završetku spoljnog programa, pod *Windows*-ima je znatno teže napraviti tako čvrstu kontrolu toka u svim režimima rada. S druge strane, takvo linijsko uklapanje spoljnog programa u tok procedure i nije baš pogodno za višeprogramski sistem kakav je *Windows*. Tako se izvođenje eksternog programa zamenjuje "lansiranjem" – startovanjem eksternog programa i bezuslovnim nastavkom glavnog toka procedure. Slično se dešava i sa menijima u proceduri: oni se definišu glavnim tokom procedure, ali se trenutak izbora opcije ni na koji način ne vezuje za osnovni tok procedure.

S obzirom na osobenosti okruženja, uveden je niz novih internih naredbi. Mnoge su jednostavne, namenjene korišćenju grafičkih mogućnosti okruženja i direktno se prevode u pozive odgovarajućih GDI funkcija. Način na koji se unose podaci takođe je mogao biti redizajniran. Za tu svrhu u program je ubačen dijalog koji se koristi za realizaciju naredbe **option**, dijalog za unos tekstualnih podataka. Nove interne naredbe mogu se podeliti na tri grupe.

Window naredbe menjaju parametre glavnog prozora programa:

- **SetTitle "string"** postavlja naziv prozora
- **Minimize** umanjuje prozor u ikonu
- **Maximize** uvećava prozor tako da zauzima ceo ekran
- **Restore** vraća prozor u normalnu veličinu

GDI naredbe menjaju grafičke parametre prozora i realizuju ispis teksta i crtanje u radnoj površini prozora:

- **SetFont "naziv fonta" veličina ("stil")** menja tekući font prema zadatim parametrima. Ispis teksta u glavni prozor nakon ove naredbe vrši se zadatim fontom. Parametar "stil" je opcion i može imati vrednost bold i italic
- **SetTextColor R-vrednost G-vrednost B-vrednost** menja boju kojom se ispisuje tekst. Boja se zadaje po principu kao kombinacija tri celobrojne vrednosti (0-255) za svaku od boja (crvena, zelena, plava). Tako niz 255 0 0 označava čistu crvenu, a niz 0 255 0 čistu zelenu boju. Crno je predstavljeno nizom 0 0 0, belo nizom 255 255 255, a srednje sivo nizom 128 128 128
- **SetBkColor R-vrednost G-vrednost B-vrednost** menja boju pozadine u funkcijama ispisa. Boja se takođe zadaje kao RGB vrednost
- **Text X Y "Tekst"** ispisuje tekst na zadatim koordinatama tekućim fontom. Koordinatni sistem počinje u gornjem levom uglu radne površine prozora
- **Rectangle X1 Y1 X2 Y2** crta pravougaonik gde je tačka X1 Y1 levi gornji ugao, a tačka X2 Y2 desni donji ugao
- **Ellipse X1 Y1 X2 Y2** crta elipsu upisanu u zadati pravougaonik. Naravno, krug se dobija zadavanjem koordinata kvadrata
- **Line X1 Y1 X2 Y2** crta liniju između zadatih tačaka

Naredbe za manipulaciju datotekama su dodate jer korišćenje odgovarajućih eksternih (DOS ovih) naredbi nije moguće kontrolisati u ovoj koncepciji programa, a njihovo odsustvo bi značajno umanjilo korisnost *WinBatch* procedura:

- **FileCopy "izvorna datoteka" "ciljna datoteka"** kopira datoteku koristeći internu funkciju za to
- **FileDelete "datoteka"** briše datoteku
- **FileRename "stari naziv" "novi naziv"** menja naziv postojeće datoteke

PROGRAM WINDOW

Vratimo se programu. *Window* procedura (listing 1) kao i obično pruža uvid u važne elemente programa. Upšteno objašnjenje upotrebe *MetaFile* mehanizma sa početka ovog teksta ovde je konkretizovano. U obradi poruke WM_CREATE, koju program dobija u trenutku kreiranja prozora, nalaze se dve bitne stvari: učitavanje teksta datoteke sa *batch* procedurom i kreiranje *MetaFile*-a. Ako se prilikom kreiranja *MetaFile*-a umesto naziva datoteke na disku prosledi NULL vrednost, kreiraće se *MetaFile* u memoriji. Kao rezultat **CreateMetaFile** funkcije dobija se identifikacija *display context*-a koji je vezan sa upravo kreiranim *MetaFile*-om. Ova identifikacija (globalna promenljiva **hDCM**) je dostupna svim ostalim funkcijama našeg programa. Upotreba kreiranog DC-a dolazi na red kada se u toku interpretacije *batch* procedure izvodi neka od grafičkih naredbi.

Za primer pretpostavimo da je iz funkcije **Execute** (izvršava naredbe procedure) pozvana funkcija **c_text** (listing 2) koja realizuje naredbu *Text*. Primetićete da se sve grafičke operacije u njoj izvode **dva** puta. Pored uobičajene promena fonta i ispisa na DC glavnog prozora, potpuno istovetne operacije se izvode po drugi put nad DC-om *MetaFile*-a. Prvo izvođenje operacije je jasno vidljivo na površini glavnog prozora, a drugo izvođenje se pamti u *MetaFile*-u.

Sada imamo kreiran **MetaFile** i imamo zapise u njemu koji predstavljaju vernu kopiju onoga što je pisano po prozoru, pa se vraćamo u *Window* proceduru našeg programa da vidimo kako se koristi ovaj zapis. Dovoljno je da pri izvršavanju programa umanjite njegov glavni prozor tako da sadržaj ne bude vidljiv, a zatim ga vratite u normalnu veličinu. Program će dobiti poruku tipa WM_PAINT i pokušati da obnovi sadržaj sada praznog prozora. Na sceni stu-



Primer reprodukcije *MetaFile*-a







Listing 1

```
HDC    hDCM;
HANDLE hMF

/*                                                                    */
/* Window pocedure                                                    */
/*     WM_COMMAND    poruke iz menija  izvrsava Batch od labele       */
/*     WM_CREATE     kreira MetaFile  ucitava Batch proceduru         */
/*     WM_PAINT      ispisuje sadrzaj MetaFile a  kreira novi MetaFile */
/*     WM_CLOSE      brise MetaFile, oslobadja memoriju               */
/*     WM_EXECUTEBATCH   izvrsava Batch od pocetka                    */
/*                                                                    */
LONG FAR PASCAL WndProc(HWND hWnd, WORD Message, WORD wParam, LONG lParam)
{
 HDC         hDC,                /* handle for the display Device     */
 PAINTSTRUCT ps                  /* holds PAINT information           */

 switch (Message)
 {
    case WM_COMMAND
         if (wParam > IDM_MENUS && wParam <= IDM_MENUS+maxopt)
         {
             /* user option   jump to a label */
             Execute( findlabel(menulabs[wParam IDM_MENUS 1]);
         }
         else
         switch (wParam)
         {
            case IDM_MENUS
                 PostQuitMessage (0),
                 break
            default
                return DefWindowProc(hWnd, Message, wParam  lParam)
         }
         break

    case WM_CREATE
         if (BatchFileName[0] == 0)
             MessageBox (hWnd  " Batch file not specified ", "WinBatch",
MB_OK)
         else
         {
             hDCM = CreateMetaFile (NULL),
             LoadBatchFile (hWnd  BatchFileName),
             PostMessage (hWnd, WM EXECUTEBATCH  0, 0L);
         }
         break,

    case WM_EXECUTEBATCH
         Execute (0)
         break,

    case WM_PAINT
         memset(&ps  0x00  sizeof(PAINTSTRUCT)),
         hDC = BeginPaint(hWnd  &ps)
         hMF = CloseMetaFile (hDCM)
         hDCM = CreateMetaFile (NULL),
         PlayMetaFile (hDC  hMF)
         PlayMetaFile (hDCM  hMF)
         DeleteMetaFile (hMF)
         EndPaint(hWnd  &ps)
         break

    case WM_CLOSE
         if ((hMF = CloseMetaFile (hDCM)) != NULL)
            DeleteMetaFile (hMF)
         else
            MessageBox (hWnd    Invalid META dc "  "Error"  MB_OK)
         DestroyWindow(hWnd)
         if (hWnd == hWndMain)
           PostQuitMessage(0)
        break

   default
         /* poruke koje se prosledjuju na default obradu  */
         return DefWindowProc(hWnd  Message  wParam, lParam)
 }
 return 0L
}
```

Window procedura

Listing 2

```
/* Postavlja parametre GDI a pred svaku GDI funkciju
void SetUserGDI (HDC hDC)
{
    if (usertextcolset)
    {
        SetTextColor (hDC  usertextcol)
        SetTextColor (hDCM  usertextcol),
    }
    if (userbkcolset)
    {
        SetBkColor (hDC  userbkcol)
        SetBkColor (hDCM  userbkcol)
    }
}

/* Ispis teksta, jednom u prozor, drugi put u MetaFile *
int o_text (void)
{
    HDC hDC
    int x, y

    hDC = GetDC (hWndMain)
    SetUserGDI (hDC)
    if (hUserFont != NULL)
    {
        SelectObject (hDC  hUserFont),
        SelectObject (hDCM  hUserFont)
    }
    x = atoi(argv[1])   y = atoi (argv[2])
    TextOut (hDC  x, y, argv[3], strlen (argv[3]))
    TextOut (hDCM  x  y  argv[3]  strlen (argv[3]))
    ReleaseDC (hWndMain, hDC),
}

/* Crtanje pravougaonika */
int o_rect (void)
{
   HDC hDC

   if (argc < 4)
       return ( 1);
   hDC = GetDC (hWndMain),
   SetUserGDI (hDC)
   Rectangle (hDC  atoi (argv[1])  atoi (argv[2]),
                   atoi (argv[3])  atoi (argv[4]))
   Rectangle (hDCM  atoi (argv[1]), atoi (argv[2])
                    atoi (argv[3]), atoi (argv[4])),
   ReleaseDC (hWndMain  hDC)
   return (1)
}
```

Procedura za postavljanje GDI parametara

Listing 3

```
SetTitle "Primer reprodukcije MetaFile a
menu
"Ispis teksta "                     PRVI TEKST
"Umanjenje i obnavljanje prozora"   DESTROY WINDOW
read
end

 DESTROY WINDOW
  Message " Prozor se prvo umanji
  Minimize
  Message   Sadrzaj prozora je sada izgubljen"
  Message   Uvecajte prozor
 RESTORE WINDOW
  Restore
  Message " Sadrzaj je obnovljen iz MetaFile a"
end

 PRVI TEKST
SetTextColor 80 80 80
SetBkColor 0 200 0
Text 10 10  "Tekst koji se ispisuje"
SetBkColor 170 170 170
SetFont "System" 12
Text 10 24   "Ova demonstracija kaže se izvrsena GDI funkcija"
SetFont "Terminal" 14
Text 10 36   pamte u MetaFile u "
SetFont "Arial  48
Text 10 52  "a posle obnavljaju
Rectangle 10 110 100 120
end
```

Primer reprodukcije *MetaFile*-a

p s svih graf čk h operac ja nad prozorom pre brisanja koji se u prazan prozor reprodukuje pozivom funkcije **PlayMetaFile**. Traka se odmotava pred vama i na prozoru se pojavljuju svi prethodni ispisi. Ovaj proces zgodno je ilustrovan *batch* procedurom koju prikazujemo na listingu 3.

Da sve ne bi bilo tako jednostavno, postoji mali problem zbog koga je deo programa koji odgovara na poruku WM PAINT nešto veći od onog što smo dosad opisali. Problem se sastoji u tome što je pre reprodukcije zapisa *MetaFile*-a taj zapis potrebno zatvoriti funkcijom **CloseMetaFile**. To znači da se u njega više ne može zapisivati, te da moramo rešiti zapisivanje i reprodukciju novih ispisa u prozor koji se mogu javiti u daljnjem toku izvršavanja *batch* procedure, tj. kao odgovor na aktiviranje neke od opcija u meniju. Kako nema sistemskog rešenja za ovaj problem, ostaje nam samo da upotrebimo jedan

TIPIČAN PROGRAMERSKI TRIK.

Kada se zatvara *MetaFile* koji sadrži dosadašnje zapise, a pre njegove reprodukcije, kreiramo novi *MetaFile*. Sada reprodukciju starog vršimo dvostruko: jednom u prozor aplikacije, a drugi put u novi *MetaFile*. Tako imamo novi *MetaFile* u kome je zapisano sve što i u starom. Al' on ostaje otvoren za dalje zapisivanje i grafičke funkcije programa ga koriste umesto starog za dalji rad. Stari iskorišćeni *MetaFile* sad možemo lepo da bacimo, to jest da ga obrišemo funkcijom **DeleteMetaFile**. Ovo žongliranje se može ponoviti neograničen broj puta u toku izvršavanja procedure – uvek će novi MF preuzimati zapis iz starog i zauzeti njegovo mesto u programu.

Što se ostatka *Window* procedure tiče, vredi spomenuti da poruka WM COMMAND, kao posledica izbora opcije u meniju, poziva funkciju **Execute** sa brojem linije iz procedure kao argumentom. Ostatak programa se uglavnom koristi tehnikama koje smo već upoznali, tako da ne zaslužuje posebnu pažnju.

Izvorni i izvršni kod programa, zajedno sa primerima *batch* procedura, može se preuzeti sa Sezama iz datoteke **\rsoft\r08\wbat.zip**. Nadamo se da će pored ilustracije *MetaFile* sistema program imati još neku zgodnu primenu.

PROGRAMSKE GRUPE

Konfiguracije i konfiguracioni programi pod DOS-om predstavljaju večitu muku; datoteke za tu namenu su potpuno nekompatibilne, mogu se menjati isključivo programom kojim su nastale i najčešće se za svaku novu aplikaciju moraju pisati iznova. U *Windows* svetu stvari stoje nešto povoljnije.

Obrad Bijelić

Konfiguracije u *Windows* programima, za razliku od najčešće prakse u DOS-u, se ne zapisuju u nekakve kriptične binarne datoteke, već se u najobičnijem ASCII obliku smeštaju u tekstualne datoteke. Ako ste nekad imali posla sa *INI* fajlovima u *Windows*-u, to je to. Kada bismo pokušali da ostvarimo istu stvar u DOS-u, naišli bismo na dosta tehničkih prepreka – zamislite samo brisanje jedne konfiguracione linije. Međutim, kada je to *Microsoft* već obezbedio, zašto bismo se vraćali nazad?

INI FUNKCIJE...

Svih šest funkcija koje u *Windows API*-ju služe manipulaciji *INI* datotekama smeštene su u *Kernel* i mogu se globalno podeliti u dve grupe. Prva služi za čitanje, a druga za pisanje *INI* datoteka, s tim da postoji još jedna podela: postoje funkcije koje operišu nad bilo kojom *INI* datotekom, kao i funkcije specijalno za *WIN.INI*, ali se one razlikuju samo utoliko što za *WIN.INI* ne treba navoditi ime – ono se podrazumeva. Deklaracije ovih funkcija izgledaju ovako:

```
int GetPrivateProfileString(app, key, default, buffer, maxsize, file)
int GetProfileString(app, key, default, buffer, maxsize)
int GetPrivateProfileInt(app, key, default, file)
int GetProfileInt(app, key, default)
int WritePrivateProfileString(app, key, string, file)
int WriteProfileString(app, key, string)
```

Podaci u svakoj *INI* datoteci podeljeni su u sekcije koje počinju imenom u srednjim zagradama. Ovaj argument (*app*) obavezan je za svaku od funkcija i pri njegovom navođenju ne treba stavljati srednje zagrade, jer to *Win* radi automatski. Drugi argument (*key*) takođe koriste sve funkcije, a služi za navođenje imena promenljive koju treba promeniti ili pročitati. Kod ovih funkcija je obezbeđena sitnica koja u praksi može da uštedi dosta truda – ukoliko na mesto ovog parametra podmetnete NULL, funkcija u bafer neće postaviti vrednost neke promenljive, već će u njega smestiti imena svih promenljivih iz odgovarajuće sekcije. Ove promenljive biće učitane jedna uz drugu kao C stringovi, pri čemu prazan string služi da označi kraj liste, odnosno smeštene su na isti način kao environment promenljive u DOS-u. Ukoliko *Win* ne uspe da nađe ono što ste tražili, on će vratiti vrednost koju navedete kao *default*.

Funkcije koje služe za čitanje celih brojeva (**Get...Int**) vraćaju naravno pročitan broj, dok ostale vraćaju broj bajtova koji zauzimaju pročitani podaci; ovaj broj može da se ograniči odgovarajućom vrednošću parametra **maxsize**, da ne bi došlo do prepunjenja nekog memorijskog bloka. Poslednji parametar je tu radi navođenja imena *INI* datoteke, pri čemu pazite da tej nastavak ne zaboravite – nema problema da se takva datoteka zove i *tetris.res*, ali možda je bolje da se poštuju postavljene *Windows* konvencije. Kompletan sistem inicijalizacionih datoteka zamišljen je i izveden odlično, sa samo jednim izuzetkom: prilikom čitanja svakog pojedinačnog parametra mora se ponovo pozivati odgovarajuća **GetProfile** funkcija. I u tome nema (za programera) ništa loše. Lošeg ima, međutim, za korisnika, jer otvaranje datoteke po desetak puta ipak troši dosta vremena. Taj problem je još izraženiji prilikom snimanja parametara, ali bi komplikacije možda bile još gore rešenje. Zamislite kada bismo pored onih 473 *handle*-ova uveli i konfiguracioni *handle*.

...I *INI* DATOTEKE

U *Windows* sistemu se inicijalizacione datoteke koriste za razne stvari, a ja sam odabrao problem koji ste verovatno uočili ako ste pravili instalaciju nekog sopstvenog *Windows* programa. Naime, ako je vaša aplikacija pravljena za *Windows*, bilo bi lepo da i instalacija bude u istom maniru. Pri tome teško da može biti nekih tehničkih prepreka – kopiranje nekoliko datoteka nije problem. Međutim, uobičajeno je da instalacioni program nakon tog kopiranja kreira posebnu grupu i u nju smesti sve module koji čine aplikaciju. Da bi se uradio ovaj posao, postoji nekoliko načina. Prvi je korišćenjem DDE (*Dynamic Data Exchange*) i komunikacijom sa *Program Manager*-om, a drugi modifikacijom *INI* datoteka.

Prvi način je prenosan, ali mi ćemo se opredeliti za inicijalizacione datoteke da bismo proučili tehnike njihove primene. Naime, *Program Manager* ima svoju *INI* datoteku (*progman.ini*), u kojoj između ostalog drži informacije o grupama koje su u njemu zastupljene. Ove informacije nalaze se u sekciji *Groups* i to kao niz promenljivih u formatu **GroupX**=Y, pri čemu je X broj grupe, a Y datoteka grupe. To nam je dalo ideju: datoteku grupe ćemo pripremiti ranije, pa ćemo je samo prebaciti u *Windows* direktorijum i u skladu s tim promeniti inicijalizacionu datoteku *progman.ini*. Ovo prvo se u listingu nalazi u proceduri *CopyGroup*, a drugo u proceduri *UpdateGroup*. Pri tome imamo samo jedan problemčić: moramo znati koji redni broj će nositi grupa, jer dve grupe ne smeju da nose isti broj. Zato ćemo na početku pročitati sve promenljive u sekciji *Groups* i zatim naći najveći broj koji se pominje u toj sekciji. Kao što je već spomenuto, dovoljno je da umesto pokazivača na ime promenljive na stek stavimo NULL i dobićemo imena svih promenljivih, u ovom slučaju to će biti sve grupe koje postoje u *Program Manager*-u.

Ostatak demonstracionog programa koristi standardne *Windows* funkcije, ali postoji jedan funkcionalan detalj, naime *Program Manager* ne zna da je datoteka *progman.ini* uopšte promenjena, a pri izlasku iz *Windows*-a on bi snimio staru verziju inicijalizacione datoteke. Za ovaj problem odabrao sam najlogičnije rešenje – jednostavno ćemo uz kontrolno pitanje izaći iz *Windows*-a, pri čemu neće biti snimljene informacije *Program Manager*-a. Ovakvo rešenje nije naročito elegantno, ali obavlja funkciju i ne predstavlja problem za korisnike. Jedino što funkcija **MessageBox** traži broj roditeljskog prozora, a naša aplikacija uopšte ne kreira nikakav prozor, zato ćemo funkcijom **FindWindow** naći prozor *Program Manager*-a i upotrebiti ga u ovu svrhu. Zaista pomalo neuobičajeno, ali je to još jedna potvrda najlepše osobine *Windows*-a – postoji mnogo puteva kojim neki posao može da se uradi.

Za razliku od ranijih primera *Windows* programa koji su bili objavljeni u "Računarima", ovaj primer se, zbog svoje male dužine, može uraditi u asembleru. Zanimljivo je koliko dugo je trebalo proizvođačima kompajlera da svoje proizvode prerade za *Windows* – verovatno je to posledica monopolističke politike *Microsoft*-a, ali činjenica je da 95% *Windows* programera koristi njihov C kompajler. Činjenica je i to da su suštinske razlike *Windows* i DOS programa na najnižem nivou ipak minimalne i da se svode na restrikcije u segmentnoj aritmetici i, naravno, *Windows API*. Da bi se jedan kvalitetan kompajler u DOS okruženju preneo na *Windows*, po mom mišljenju, jedini problem je specifičan (ali lak za realizaciju) linker, ostatak je obezbeđivanje što bolje podrške *Windows API*-ju i sve što uz to ide. Prilično sam iznenađen što *Nantucket* svoj *Clipper* nije već preradio za *Windows*, jer mi se čini da bi takav potez *Clipper* programeri toplo pozdravili.

No, vratimo se primeru. Treba primetiti da se slobodno mogu pozivati standardne DOS funkcije za pristup datotekama. Razlika u odnosu na DOS nema baš mnogo – početni kod je malo drugačiji nego kod DOS-a, nema postavljanja segmentnih registara, jer je DS automatski postavljen na DGROUP. Prenos parametara sistemskim (i svim ostalim) funkcijama ide preko steka i strogo koristi paskalsku konvenciju: parametri se na stek ređaju onim redom kako su navedeni i funkcija koja je pozvana mora sama da ih skine sa steka. Ako pogledate neki *Windows* program, primetićete da jako mnogo koda ode na smeštanje parametara

Listing 1

```
[illegible]
```

Program za kreiranje grupe za *Windows* aplikacije

na stek. Zato preporučujem da obavezno koristite instrukcije procesora 80286 (pogotovo sa *Turbo Assembler*-om) koji ima instrukciju za smeštanje konstanti na stek, tzv. **push immediate**. Ukoliko je ta konstanta osmobitna vrednost, odgovarajuća **push** instrukcija zauzeće samo dva bajta (ako je šesnaestobitna, onda tri). Ako je baš neophodno da program može da se izvršava i na XT-u, a pri tome koristite *Turbo Assembler*, direktiva SMART na početku programa rešiće vas muka prepravljanja čitavog programa.

NASTAVAK SA STRANE 77

trivijalne ali često upotrebljavane operacije odabira manjeg odnosno većeg broja od dva prosleđena. **View** poziva svoja imenjakinje u objektima tipa **TLine**, određuje granice prevougaonog područja u kome je poligon, izračunava parametre pripadajuće ravni i sinus ugla koji ona zaklapa sa negativnim krajem x ose. (Kod nalaženja jednačine ravni uzimaju se početne tačke prve tri duži poligona i uvrštavaju u izvedene formule. Obavezno je da tačke budu nekolinearne: metoda **View** to ne proverava pošto se sva tela i svi poligoni oblikuju od strane pet klasa za generisanje primitiva, koje i ne dopuštaju takav slučaj. Ipak, ako nameravate dopisivati vlastite klase za definisanje tela – na primer rotiranjem ili izvlačenjem kontura – bilo bi mudro ubaciti i proveru nekolinearnosti. Još bolje, već u samom procesu nastajanja tela i poligona se može ugraditi preventiva tako da se dve susedne duži spoje u jednu ako se otkrije egzistencija zajedničke prave-nosača.) Metoda **Intersection** je funkcija koja će vratiti vrednost *True* ako je prosleđena joj tačka (Px,Py) unutar perspektivne projekcije poligona, pri-tom će polje **Dist** sadržavati udaljenost od posmatrača. **Show** i **Move**, polimorfne metode, samo razašilju svojim dužima primljene poruke o prikazivanju i pomeranju poligona. **AppLine** inicira i dodaje novi **TLine** objekat na rep liste, i kao poslednja metoda, destruktor **Done** garantuje da će pre ukidanja objekta biti iz memorije oslobođena i pripadajuća lista duži.

KLASA TBody

Isto kao što objekat **TPoly** klase drži za sebe privezanu listu TLine objekata, tako i svaki primerak **TBody** objekta ima sopstvenu listu **TPoly** poligona. Otuda proističu njihove zajedničke osobine koje obezbeđujemo nasleđivanjem. Nama uočili se da neka metode iz **TPoly** nisu, sa stanovišta samo te klase, napisane najracionalnije: napisane su tako da odgovaraju toj klasi ali i svim drugim koja se iz nje izvode. Takođe, ni neke nove metode iz **TBody** nisu namenjene samo njoj nego i klasi TGroup koja se na nju nastavlja.

Objekat klase **TBody** je najniži u hijerarhiji kome korisnik aplikacije pristupa neposredno. Za tu svrhu mu se dodeljuje identifikator u vidu pakovanog stringa dužine 10 karaktera i ekstenzije dužine 4 karaktera. Sadržaj prvog stringa određuje korisnik po svom nahođenju, a ekstenziju dodaje konstruktor metoda neke od pet naslednčkih klasa za građenje tela. Prikazivanje tih naziva u tekst modu povereno je metodi **List.**

Za polimorfne metode **View**, **Show**, **Intersection** i destruktor **Done** važi otprilike isto što i za istoimene metode iz prethodne klase. **CmpDistance** prvo svakom primerku objekta poligona izvisave **Intersection** funkciju, a zatim, ako je vraćeno tačno, poredi njegovu udaljenost od tačke gledišta sa dotad zabeleženom najmanjom vrednošću. Preko zastavice **Inter** joj se saopštava da li je već zabeleženo da je neki objekat „pogođen" zrakom svetlosti ili nije. Tako **CmpDistance** zna da li će pri prvom sledećem pozitivnom odgovoru funkcije **Intersection** izvršiti poređenja ili će, ako je **Inter** bio **False**, dobijenu udaljenost uzeti za početnu. Njena uloga u procesu senčenja je da iz neke od lista objekata izdvoji onaj najbliži posmatraču i da zapamti sinus upadnog ugla. **CmpDistance** se koristi bez izmena i od strane naslednčkih klasa **TGroup**. (Da bi se ispoštovala kompatibilnost tipova morao se vaznik **TPoly.Next** iz prethodne klase deklarisati kao običan pointer.) Konačno, **Triangle** i **Square** u sadejstvu sa nasleđenom metodom **TPoly.AppLine** dodaju u listu poligona po jedan trougaonik ili četvorougaonik koji najčešće koristimo pri formiranju tela.



ENGLESKO-SRPSKI/SRPSKO ENGLESKI REČNIK ZA PC RAČUNARE

Ranka Jovanović

Autor: Milorad Bešlin
Izdavač: „MB SOFT" Novi Sad

Svako ko se na bilo koji način bavi stranim jezicima zna da je dobar, pregledan i pouzdan rječnik apsolutno neophodna, ali i veoma rijetka stvar. Kao prevodilac „od karijere", dijelim neprijatno iskustvo autora ovog PC rječnika, koji kaže da „po već legendarnim Marfijevim zakonima, potrebno vrijeme da se pronađe željena reč raste sa njenom važnošću za nas u datom momentu." Osim toga, što je rječnik bogatiji riječima, to je veća vjerovatnoća da ćemo naći željeni odgovor, ali i za traženje treba nam više vremena i strpljenja. Najčuveniji američki izdavač rječnika Merriam-Webster je davno pokušao da svoj najveći rječnik engleskog jezika izda u što manjem formatu. Štampan je na najtanjem papiru i najsitnijim slovima, tako da se prodavao s povećalom da bi se mogao čitati. Ipak je bio toliko glomazan da je kupcima nuđen i specijalni stočić na koji se mogao osloniti prilikom listanja.

Srećom, PC računari na sebe preuzimaju sve više dosadnih, dugotrajnih, pa i fizički teških poslova (u koje spada i skidanje dobrih velikih rječnika sa police). Naravno, pod uslovom da postoji odgovarajući program. Odavno su rječnici poznatih anglo saksonskih izdavača ugrađeni u popularne tekst procesore, ali je to donijelo olakšanje samo odličnim poznavaocima engleskog jezika, za tekstove pisane isključivo na engleskom. Znači, još uvijek ništa od prevođenja na srpski i sa srpskog.

Sada se i za to pobrinula novosadska firma MB SOFT, odnosno autor Milorad Bešlin. Srpsko-engleski/englesko srpski rečnik za PC računare (verzija 2.0), sa fondom od oko 90.000 riječi, dolazi u paketu od 4 diskete i sa uputstvom za upotrebu. Instalacija je brza i jednostavna, pod uslovom da na disku imate 3.5 megabajta raspoložive memorije.

Nakon instalacije se ulazi u vizuelno lijepo riješen plavo bijeli meni (sa crvenim detaljima) koji nudi opcije Engleski, Srpski, Unos, Izmene, Brisanje, Kraj. Postoji i sistem pomoći korisniku u koji se ulazi sa F1, nakon čega se otvara prozor sa opcijama Pomoć, Gramatika, Fajl, Izlaz.

Osnovna funkcija rječnika je pretraživanje koje se vrši po srpskim ili po engleskim rečima. Recimo, nakon izbora opcije „engleski" u invertno osvijetljeno polje se unosi engleska riječ čiji prevod tražimo, a zatim se pritisne ENTER. U desnom uglu ekrana se pojavljuje prozor i u njemu sve srpske riječi koje su prevod date engleske riječi. Ako tražena riječ ne postoji ili je pogrešno napisana, otvara se veći prozor sa abecednom listom riječi koje su joj po ortografiji najbliže. Dalji pregled ove liste vrši se pritiskom na tastere PgUp ili PgDn. Brzo i jednostavno.

PC rječnik predviđa i mogućnost unosa novih riječi koji je ograničen samo slobodnim prostorom na vašem disku. Autori navode da se u svaki slobodan megabajt može unijeti oko 70.000 novih riječi! Moguće su i ispravke i izmjene unesenih riječi, kao i brisanje.

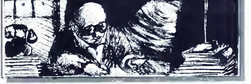
Posebna pogodnost je to što je moguć uvoz fajla u ASCII kodu, a zatim prevod odabrane riječi bilo na srpski ili engleski. Riječ se odabira postavljanjem kursora na njen početak i pritiskom na taster F2 (engleski) ili F3 (srpski). Ukoliko riječi nema u rječniku, pojavljuje se lista riječi najbliskijih po ortografiji. Tako da se dalje može pretraživati pomoću PgUp ili PgDn.

Dakle, opet lako i jednostavno, programerski dobro zamišljeno i izvedeno. Ali već pri prvom pregledu rječnika pokazuje se da stručnom dijelu ne možemo uputiti istu pohvalu. Metodom slučajnog uzorka smo odabrali pedesetak termina i zapazili iznenađujuće mnogo grešaka. Ponovljene probe su, na žalost, samo potvrdile prvi utisak – bilo bi potrebno mnogo unosa, ispravki i brisanja da bi se sve dovelo u red. Naše zamjerke se mogu svesti na sljedeće:

1) slovne greške, vjerovatno nastale nesmotrenim unosom, npr. *scean* = skenirati (naravno *scan*), *nobodi* (umjesto *nobody*), *factious* = frakcionaški (umjesto *fractious*), *friga* (umjesto *fridge*) itd.;
2) brkanje vrsta riječi – za srpsku imenicu kao prevod se daje engleski pridjev ili glagol, npr. neprijatelj = *hostile* (što u stvari znači neprijateljski), *accept* = prijem (u stvari primiti, dok je prijem *acceptance*),
3) pogrešan prevod, npr. *personal* = pokretan, pokretnost (?), *above* = na dalje nego (?)
4) liste ponuđenih prevoda ne daje prioritet najfrekventnijem značenju, te se ono često nađe na samom začelju ili se uopšte ne pominje. Na primjer, daleko najčešće značenje riječi *abroad* je *inostranstvo*. Međutim, rječnik će vam ponuditi čitav niz neobičnih pa i arhaičnih rješenja, ali ne i ono pravo: napolje, u tuđini, iza granice, za granicom, van kuće, spolja, široko, na sve strane, u zabludi (!?)

Kada dobijete *Rječnik za PC računare* ili kraće „PC rječnik" zar ne očekujete da ćete imati pomoć i u prevođenju barem najosnovnije računarske terminologije? E, ovdje ćete saznati da je *hardware* samo gvožđurija, *keyboard* samo i jedino klavijatura, *monitor* je oklopni brod, starješina ili stariji učenik (u školskoj terminologiji *monitor* u stvari znači redar), a ako pokušate naći *software, buffer, bus* ili, ne daj bože, *downloading*, otvoriće se prozor sa pogledom u prazno. Takve i slične sitnice nisu predviđene – da li zato što autori uzimaju zdravo za gotovo da je ova terminologija neprevodiva ili zato što smatraju da vlasnik PC računara sve to već treba da zna? Doduše, opet vam ostaje opcija Unos, pa na posao. Možda će se na to odlučiti neki strpljiv korisnik kojem će ovaj softver poslužiti za pravljenje ličnog, usko stručnog rječnika.

Sa stručnog stanovišta, najlošiji dio je gramatika, za koju se kaže da obuhvata „elementarne pojmove gramatike engleskog jezika". Svako ko je iole upućen u engleski jezik odmah će prepoznati bisere poput onog da je zamjenica za drugo lice jednine *thou, thy, thine* (jeste, ali kad bismo se vratili u Šekspirovo doba), da u engleskom postoje glagolska vremena koje se zovu „sadašnje svršeno vrijeme" *(Present Perfect!)*, „neodređeno (opšte) vrijeme" *(Present Simple!)* i „opisna vremena" (?). Čak i ako pređemo preko ovog „slobodnog" prevoda, dolazimo do primjera za navedena vremena – tu tek nastaje potpuna zbrka od mnoštva pogrešnih ili pobrkanih jezičkih oblika. Ukratko, ni manje gramatike ni više netačnih podataka. Bojimo se da tu nema pomoći od izmena i dopuna, već jednostavno treba sve zaboraviti i izbrisati, pa uz pomoć relevantne literature i stručne savete pisati ponovo.



PROBLEM CELOBROJNOG CENTRA

Najavili smo da je Pitalica teška, ali nismo znali da je *toliko* teška! Možda stvar i nije bila u težini problema, nego u utisku da će biti potrebno silno računarsko vreme da se on reši. U stvari, moderni personalni računari nas ponekad zapanje svojom brzinom...

Podsetimo se pre svega postavke. Na slici 1 prikazan je oštrougli trougao ABC čije se visine AD, BE i CF seku u ortocentru H. Rastojanja AE, AF, BF, BD, CD, CE, AH, BH, CH, DH, EH i FH izražena u centimetrima su međusobno različiti celi brojevi. Trebalo je odrediti stranice trougla tako da njegova površina bude minimalna.

MINIMALNA POVRŠINA TROUGLA

Primetimo, pre svega, da je oštrougli trougao ovim visinama podeljen na 6 pravouglih trouglova koje možemo da grupišemo u tri pare sa zajedničkim hipotenuzama AH, BH i CH. Posmatranjem jednog od tih parova (na primer, onog čija je hipotenuza AH) uočavamo da važi

$AH^2 = AF^2+FH^2 = AE^2+EH^2$

pri čemu su sve navedene dužine različite – redovni čitaoci ove rubrike primetiće da je ovaj deo problema zapravo naša 70. Pitalica! Trebalo je, dakle, generisati sve „duple" parove Pitagorinih brojeva proveravajući da li je ispunjen uslov o različitosti i, ako jeste, određivati površine tragajući za minimalnom. Teorija je, dakle, sasvim jednostavna i kratka, a praksa može i da potraje – zavisno od autorove programerske veštine, program će se izvršavati nekoliko minuta ili nekoliko dana. Ovo je očito „uplašilo" mnoge rešavače pošto je izgledalo da je tako obimno istraživanje solidan zalogaj i za neki VAX ili IBM 370. No, ne mora da bude baš tako – program **Miloša Pivulovića** za svega svega 11 minuta Amiginog vremena (MC 68000 na 7.14MHz, bez koprocesora) pronalazi optimalno rešenje zadatka – stranice trougla su 3000, 1675 i 2925, a minimalna površina 2632500. Ako su duži ne izražene u centimetrima, površina trougla bi bila 263.25 m² – ne bi lako stao u sobu! Program je nešto duži nego što je uobičajeno u ovoj rubrici pa ga zato objavljujemo samo na Sezamu, gde se može pronaći u direktorijumu **reoff**. Program je inače preuzet iz časopisa *Personal Computer World*.

U predviđenom roku (u „Računarima 84" je štamparskom greškom pomenut 1. septembar umesto 1. oktobre) smo primili svega četiri optimalna rešenja Pitalice, jedno pogrešno i 6 neoptimalnih. Pošto smo pažljivo pregledali sva četiri, zaključili smo da se radi o dobro obrazloženim i majstorski pisanim programima, pa je Fortuna odlučivala o nagradama. Tako je **Petar Marjanović** iz Zemuna dobio 20.000 dinara, **Biljana Srdanov** iz Žagubice 15.000, a **Boško Koprivica** iz Sremčice 10.000 dinara. Nagrađenima čestitamo, baš kao i autoru četvrtog korektnog rešenja **Milošu Pivuloviću** koji je „ostao ... u bubnju".

OPROŠTAJ OD EUREKE

Iskoristio bih ovu priliku da dodam još nekoliko reči namenjenih onima koji su osim Pitalica rešavali i probleme iz rubrike „Eureka" koja je nekih šest i po godina (od februara 1986) izlazila u časopisu „Galaksija". Posle dve decenije izlaženja „Galaksija" se na žalost u septembru i oktobru nije pojavila na kioscima. To je kulminacija dugogodišnje krize u kojoj se ovaj jedinstveni časopis za popularizaciju nauke nalazio. Kraj „Galaksije" je prirodno i kraj „Eureke", ali je kod rubrika ovoga tipa pomalo neprijatno što „ostaje u igri" nekoliko zadataka čija rešenja nisu nigde objavljena. Kada mi se to prošli put desilo (nekada jako davno, još pre „Računara" u „Galaksiji" je postojala mala rubrika „Nagradni zadatak", neka vrsta praroditelja Pitalica i Eureke), dva zadatka čija rešenja nisu objavljena „pratila" su me godinama (ne verovali je koliko dobro pamte ljubitelji matematike) i „morao" sam da demonstriram njihovo rešavanje (koje nije bilo ni malo jednostavno pošto se po Marfiju moralo „potrefiti" da izostanu baš rešenja prilično teških zadataka) na puno mesta, sve dok te probleme nisam „reprizirao" u ovoj rubrici i tako „odradio dug".

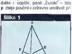
Da se to ne bi ponovilo i sa „Eurekom", rešenja četiri problema iz „Galaksije 240" (maj 1992) nalaze se u „Galaksiji 242", koja se može *download*-ovati sa Sezama, a na sličan način će biti pristupačna i rešenja problema iz „Galaksije 241" (jun 1992). Čitaoci „Eureke" koji nemaju računar i modem ili pristup Sezamu a žele da imaju rešenje nekog od ovih problema, mogu mi na adresu „Računara" poslati na sebe adresirani koverat sa dovoljnom količinom poštanskih maraka i – dobiće fotokopije rešenja. Hvala svima koji su rešavali i predlagali zadatke i, uopšte, pisali „Eureki" – bilo je zbilja poučno i zabavno uređivati je!

ŽIL VERNOV PROBLEM

Volite li Žil Vernove romane? Jedan od njih (za sada nećemo reći koji) je očito inspirisan slavnom pripovetkom Edgara Alana Poa „Zlatni jelenak" – glavni junak tog romana je u mladosti osuđen na smrt zbog krađe dijamanata i ubistva vojnika koji su sprovodili pošiljku (što, naravno, nije učinio), pobegao je, proveo čitav život u susednoj državi pod lažnim imenom i, najzad, sticajem okolnosti, bio prinuđen da se vrati na mesto zločina i suoči sa presudom. Jedina nada mu je bilo pismo u kome je pravi zločinac, mučen grižom savesti, pred smrt priznao svoje (ne)delo, ali je to pismo na žalost bilo šifrovano a čovek koji je znao šifru mrtav. Sudija Žarikes koji se zainteresovao za čitav slučaj je na neki volšeban način (to je slaba strana romana) „pogodio" kako je poruka šifrovana, ali za razliku od Vilijama Leguanda iz „Zlatnog jelenka" nije bio dovoljno sposoban kriptoanalitičar da bi odredio ključ i tako dešifrovao poruku. Na kraju je do ključa došao na prilično „neortodoksan" način. Da vidimo da li će naši čitaoci biti uspešniji.

Šifrovana poruka data je na slici 2, dok postupak šifrovanja ilustruje slika 3. Ispod otvorenog teksta ispisuje se višecifreni broj (u primeru sa slike 3 to je broj 7694) a onda se svako slovo „pomera" po abecedi (abeceda obuhvata slova čćđšž kao i qwxy) za onoliko mesta koliko pokazuje cifra napisana ispod tog slova (ako se u brojanju dođe do Ž, prelazi se prirodno na A) – tako nastaje šifrat. Znajući ovu metodologiju, treba da dešifrujete poruku sa slike 2 i ujedno odredite ključ šifre, tj. korišćeni višecifreni broj.

Slika 1



Odrediti stranice oštrouglog trougla tako da njegova površina bude minimalna

Slika 2

[illegible]

Šifrovana poruka

Slika 3

Otvoreni tekst: [illegible]
Ključ: [illegible]
Šifrat: [illegible]

... i postupak šifrovanja u Pitalici 79

KAKO DOSTAVITI REŠENJE

Obrazložena rešenja Pitalice, zajedno sa programima koje ste napisali, pošaljite na uobičajenu adresu „Računari" (za Dejanove pitalice), Bulevar vojvode Mišića 17, Beograd, tako da prispeju **pre** 25. novembra 1992. Sva pisma sa korektnim rešenjima konkurišu za novčane nagrade od 32.000, 24.000 i 16.000 dinara, dok će kuponi (ili njihove fotokopije) na koje je upisan identifikacioni broj učestvovati u godišnjem takmičenju rešavača Pitalica. Identifikacioni broj dobijate tako što u prve tri kućice upišete **poslednje** tri cifre nečijeg broja telefona, u sledeće dve godinu vašeg rođenja, a na kraj dodate dve cifre po izboru. Obratite pažnju da sva rešenja koja šaljete u toku godine nose isti identifikacioni broj. Osim obićnom, rešenja ove Pitalice možete da pošaljete i elektronskom poštom posredstvom Sezama – koristite ime REDAKCIJA PITALICE REŠENJE ... iz komandnog moda pošaljite ličnu poruku korisniku pitalice (*mail write pitalice*).

79: Ključ je broj: ______

Ime i prezime ______

Adresa ______

Mesto ______

Kompjuter ______ Vreme ______

Identifikacioni broj |__|__|__| |__|__| |__|__|

iz broja tel. — god. rođ. — po izboru

Koja verzija?

Jedna od najboljih usluga koju sebi možete učiniti pri razvijanju iole složenijih programa je razbijanje projekta na što manje module, uz logično grupisanje funkcija (prema nameni, prema osetljivosti na okruženje kada je prenosivost programa bitna i slično). Bio bih sklon da tvrdim kako je dobro čak i mehanički primeniti jednostavno pravilo da je svaki program koji preti da pređe hiljadu redova zreo za rastavljanje na module. Ipak, mnogi to ne čine jer je potreban određen trud da se ova podela napravi – treba pre svega napraviti **makefile** datoteku, jer je prevođenje projekta sastavljenog iz modula bez *Make* programa praktično nemoguće. Oni koji ne znaju ili nemaju običaj da koriste *Make* i ne slute šta propuštaju, ali o ovoj temi je već bilo reči u „Računarima" pa da je sada ne otvaramo ponovo.

Ukoliko ne razvoju nekog projekta radite dugo, puštajući na testiranje ili u upotrebu radne verzije, sigurno ste se suočili sa problemom vođenja evidencije. U nekom trenutku ste program dali jednom korisniku, potom ste nastavili razvoj, sledeću verziju dali drugom korisniku na testiranje, zatim je prvi pronašao neki bag koji je zahtevao silnu intervenciju u jednom od modula... nakon nekoliko dana (meseci, godina?) razvoja više nećete znati koji korisnik o kojoj verziji govori i na šta se njegove primedbe odnose.

Naravno, neku evidenciju morate da vodite – najčešće je dovoljno u sam kod ubaciti kratku napomenu sa datumom kada je izmena izvršena. Time se ipak ne rešava problem različitih verzija koje su date na testiranje – potrebno je i program numerisati jedinstvenim brojem verzije. Alternativno bi se mogla voditi odvojena evidencija izmena po principu „oznaka verzije – šta je u njoj menjano". Nakon toga bi snalaženje bilo koliko-toliko olakšano.

Napisati kratak komentar svaki put kada izvršimo izmenu zaista nije veliki posao, ali je numerisanje svake sledeće verzije prilično zametno, jer se jednostavno ne sme izostaviti. Ako vam ne pređe u naviku da nakon svake izmene pozovete i modul u kome je upisan broj verzije i njega promenite, čitav sistem evidencije pada u vodu.

Na ovaj problem praksa me je često podsećala, pa sam se potrudio da pronađem pogodan način koji ne bi zahtevao moje angažovanje oko održavanja broja verzije. Oslanjanje na DOS vremenski pečat koji nosi **exe** datoteka je prilično nepouzdano. Logična ideja je da se po nekoj automatici pri svakoj izmeni bilo kog modula programa, celom programu dodeljuje nova oznaka verzije.

Može se lako zamisliti i neko jednostavno rešenje – kao deo projekta dodali jedan modul u kome bi bila upisana oznaka verzije, recimo modul VERSION.C sa samo jednom linijom

int VERSION = 0;

Zatim bi trebalo napisati program, neka se zove INCVER, koji čita ovu datoteku i kreira novu sa brojem verzije za jedan većim. U **makefile** bi se potom dodalo da se pri svakom kompajliranju prvo startuje INCVER i potom prevodi i VERSION.C, čime bi u **exe** automatski bio ugrađivan novi broj verzija.

Rešenje jeste jednostavno, ali mi ipak nije delovalo elegantno – prosto je morao postojati jednostavniji način. I zaista, ako se umesto broja kao oznaka koriste datum i vreme kada je **exe** nastao, onda odlično mogu da posluže predefinisani makroi koje poznaje MSC – __TIME__ i __DATE__. Ovi makroi se u trenutku prevođenja programa zamenjuju stringom – ascii oblikom trenutnog datuma i vremena. Možemo i dalje zadržati koncept odvojenog modula u kome je oznaka verzije, sada sa sledećim sadržajem:

char *VERSION = __DATE__ " " __TIME__;

Sve što treba da obezbedimo je da **make** uvek prevodi i ovaj modul, nezavisno od toga da li je menjan, i **exe** fajl će uvek imati ažurno zapisan trenutak nastanka.

Ipak, kod složenih projekata ni sa ovim podacima snalaženje nije baš lako. Bilo bi zapravo najbolje da možemo u svaki modul upisati oznaku lokalne verzije za taj modul, napuštajući koncept vođenja globalne oznake verzije. Na prvi pogled ništa lakše – umesto posebnog VERSION.C, u svaki programski modul na početku dodamo liniju sa __DATE__ i __TIME__ makroima. Na žalost, pojavljuje se problem. Prvo, za svaki modul bi trebalo da izaberemo poseban naziv promenljive, što onda nameće problem kako takvu informaciju ispisati. Ako imamo promenljive MODUL_A_VER, MODUL_B_VER itd, to znači da bi trebalo napraviti posebnu rutinu za ispis, recimo

```
int showversion()
{
  puts(MODUL_A_VER);
  puts(MODUL_B_VER);
  .. itd
}
```

Ni ovo nije posebno teško (mada još više destimuliše deljenje izvornog koda na module, jer se treba setiti da se pri svakoj daljoj podeli ažurira i ova funkcija), ali koncept ima jednu veliku manu. Zamislite da projekat realizujete tako što pravite biblioteku vaših funkcija koja služi kao osnova za generisanje više izvršnih programa. Ako u neki od tih programa ubacite poziv **showversion()**, ustanovićete da će kompletna biblioteka biti povezana u **exe**, čak i ako je samo deo funkcija iz nje potreban. Razlog je, naravno, u tome što funkcija **showversion()** referiše tu jednu promenljivu iz svih postojećih modula, čime se forsira njihovo ubacivanje u **exe**, iako se možda ništa od samog izvršnog koda iz modula ne koristi.

Listing 1 SHOWVER.C

```
//
// SHOW VERSION OF MODULES IN BIN FILES
// Zoran Zivotic  Računari  10 92
// compile MSC 7 00  cl -Ox showver + \c700\lib\setargv
//
#include <stdio.h>
#include <string.h>

static char *__ver_id = "$>z<Z" __FILE__ " " __TIMESTAMP__;

#define BUF_SIZE 0x8000

char buf[BUF_SIZE];
long pos;
int lim ;
FILE *f;

int scan(void)
{
  int i;

  for(i=0;i<lim-64;++i)
    if (buf[i] != '$')
      continue;
    else if (memcmp(buf+i+1 ">z<Z",4))
      continue;
    else { i+=5; putchar('\t'); puts(buf+i); }
}

main(int argc, char *argv[])
{
  for(j=1 j<argc;++j)
  {
    puts(strupr(argv[j]));
    if ( (f=fopen(argv[j],"rb")) == NULL )
    {
        printf("\nFile %s not found\n",argv[j]);
        continue;
    }
    pos = 0L;
    for(;;)
    {
      fseek(f,pos-64,SEEK_SET);
      if ( (lim = fread(buf,1,BUF_SIZE,f)) == 0 )
        break;
      pos += lim;
      scan();
    }
    fclose(f);
  }
}
```

Sada nastaje interesantna situacija – imamo zgodan način da modul numerišemo, ali ne i da tu informaciju prikažemo. Ovde mi je konačno palo na pamet da nema razloga da se prikaz verzije obavlja u samom **exe** programu – za to možemo napraviti mali programčić koji će generalno obavljati taj posao. Dakle, napravimo **showver.exe** i potom sa

showver mylib.lib, ili
showver myexe.exe

dobijamo listu svih modula koji se nalaze u programu, sa datumom i vremenom kada su nastali. Da bi ovo bilo moguće, dovoljno je da u VERSION promenljivu dodamo neki marker koji bi služio za identifikaciju, recimo

char *VERSION = "$>z<Z" __DATE__ " " __TIME__;

Sve što nam sada treba je program koji će pretražiti exe datoteku (lib, obj ili bilo kakav drugi format!) i ispisati tekst iza svih pojava markera "$>z<Z". Naravno, možete izabrati bilo šta drugo, pa i kraći marker, jer tačno prepoznavanje markera nije posebno kritično – ako program slučajno sadrži na nekom mestu izabrani niz, dobićemo i jednu suvišnu liniju sa „besmislenom" informacijom, ali od toga nikakve štete.

Rešenje ima i dodatnih prednosti. Pošto naziv promenljive koja nosi oznaku verzije više nije bitan, možemo koristiti uvek isto ime, dodeljujući mu atribut static kako bi bila vidljiva samo unutar modula. Linije će, dakle, u svim modulima biti ista, pa se može napraviti u obliku makroa

#define VERSION static char *__ver_id = "$>z<Z" __DATE__ " " __TIME__

Nakon ovoga, sve što treba da uradimo je da na početak svakog modula dodamo liniju

VERSION;

i imaćemo, bez ikakve dalje intervencije, potpunu evidenciju kada je svaki od modula nastao.

Upotreba makroa __DATE__ i __TIME__ ima jednu manu: njihove vrednosti će se menjati svaki put kada prekompajliramo modul, čak i ako ga nismo menjali, i ako je kompilacija rezultat eventualne izmene neke globalne header datoteke. U ovoj koncepciji je zato daleko pogodnije koristiti drugi makro iz ove grupe – __TIMESTAMP__ se zamenjuje kompletnim datumom i vremenom, ali onim koji nosi datoteka na disku. Ako uz ovo dodamo i __FILE__ koji se zamenjuje nazivom same datoteke, onda dolazimo do konačnog rešenja

#define VERSION static char *__ver_id = "$>z<Z" __FILE__ " " __TIMESTAMP__

Ostaje nam još samo da napravimo program za prikazivanje ovih podataka. Jedna skromna varijanta je data na slici – može se, naravno, poboljšati sa nekoliko prekidača kojima bi se opciono zahtevalo da spisak modula bude sortiran po vremenu menjanja, ili da se podaci o modulima iz jedne datoteke porede sa podacima iz druge i daje lista samo onih modula koji su različiti. Ukoliko program **showver** linkujete sa datotekom **setargv.obj** iz **lib** direktorija, jednostavno dobijate mogućnost da kod startovanja navedete naziv datoteke sa džoker znacima. Tako će jedno **showver *.exe** dati listu svih modula u svim programima koji čine vaš složeni projekat.

530 [YUZI] Poslednjih dana septembra dobili smo novu kodnu tabelu prilagođenu srpskom jeziku. Prikazana je na slici 1 i, kao što se vidi, nosi oznaku ISO I.B1.015. Primetimo da su mesto u novoj tabeli "zaslužila" sva slova srpske ćirilice kao i karakteristična latinična slova čćđšž, da je kompletna grafika zadržana na originalnim mestima i da su, na nekoliko preostalih pozicija, ubačena neka grčka slova i drugi znaci. Leva polovina kodne tablice popunjena je nadrinulim ASCII setom, s tim što je standardom predviđeno da se na pozicije 40, 5B-5E, 60 i 7B-7E mogu staviti i latinična slova Ž Š Đ Ć Č ž š đ ć i č respektivno, čime se po potrebi održava kompatibilnost sa široko rasprostranjenim YUSCII rasporedom.

Novi standard, koji je Savezna komisija usvojila posle veoma opsežnih priprema i višemesečne diskusije (pored ostalog i na Sezamu), ima nekoliko veoma jakih strana. Pre svega, uspelo se da ćirilica bude kontinualna, pogodna za konverzije velikih slova u mala, sortiranje i tome slično. Grafika je ostala na originalnim mestima što znači da će velika većina komercijalnih programa savršeno raditi – ćirilica je "pokrila" strane i grčka slova i pojedine matematičke simbole. Najzad, latinična slova čćđšž postavljena su takođe kontinualno i na logičnim pozicijama (mala slova su bezmalo po rasporedu "Računari"). Ukratko, standard dobro prilagođen PC računarima koji rade pod DOS-om. Što se mana tiče, pažljivi pogled otkriva neke nepotrebno uvedene nedoslednosti. Pre svega, znak "alfa" je stavljen na poziciju A5 iako nije bilo nikakve prepreke da se ostavi na poziciji E0 gde je "bez veze" stavljeno slovo "mikro" – pošto već "mikro" nije moglo da ostane na svojoj poziciji, ono je moglo da bude stavljeno na A5, a da alfa ostane na svom mestu. Slična priča i za *bullet* koji je mogao ostati na "prirodnoj" (ili bar "srodnoj") poziciji A7, umesto što je tu stavljeno slovo "omega" koje je sasvim lepo moglo doći na sadašnju potpuno proizvoljno izabranu poziciju *bullet*-a AD. Diskutabilan je, osim toga, i izbor znakova koji će ostati u setu karaktera – uvršćene su neke prilično nepotrebne stvari kao "španski navodnici" na AE i AF, a izbačeni neki univerzalno potrebniji znaci kao pun *bullet*, plus/minus ili možda "kockica" sa FE. Sa koncepcijske tačke gledišta, mnogi diskutanti u raspravi su smatrali da nema potrebe za odvajanjem ćirilice i latinice i da bi za obe mogli da se koriste isti (ASCII) kodovi, s tim što bi se njihov ispis rešavao izborom odgovarajućeg fonta. Najzad, standard je neprilagođen grafičkim okruženjima kao što su *Windows* ili *Ventura*, ali zbilja nije ni lako predložiti raspored koji bi se tu "prirodno" uklopio.

Ukratko, dobili smo novi standardni raspored domaćih slova. Videćemo koliko će zaživeti u praksi; oni kojima se dopadne naći će sve potrebne alatke (drajver za tastaturu, VGA fontove itd) na Sezamu, a kada su već tamo, mogu i prokomentarisati standard u konferenciji SEZAM, tema *yu.kod*.

531 [OTXT VENTURA, PC] Tabelu sa slike 1 nije bilo ni malo lako napraviti u *Venturi*. Najveći problem je kombinovanje slova – na istoj slici treba imati ćirilicu, latinicu i grafičke znake koji normalno nisu deo seta karaktera koji *Ventura* podržava. Obzirom da *Bitstream* (još?) ne proizvodi ćirilične fontove, koristili smo Helvetiku koju je dizajnirao *Zoran Životić*. Zatim smo sintetisali fontove koji u donjem delu ASCII tablice (32-127) imaju znakove koji se normalno nalaze u gornjem delu PC-jevog karakter seta (128-255). Tako smo, recimo, u polje C0 upisali apostrof a onda sa SET FONT taj apostrof prebacili u drugi font u kome se on ispisuje kao odgovarajući grafički karakter. Srećom, manuelno obeležavanje fontova u svakom od polja se moglo izbeći primenom odgovarajućeg makroa pomoću koga je tekst editor dodao potrebne prefikse i postfikse. Ukratko, poprilično posao.

Usput smo primetili i jedan *bug Ventura 3.0*: u polja tabele se ne može upisati usamljena zapeta. Tačnije rečeno, može se upisati, može se i odštampati, ali posle snimanja teksta na disk i njegovog ponovnog učitavanja nastaje totalni haos: sve sledeće kolone se pomeraju za po jedno mesto ulevo, potpuno uništavajući tabelu. Jedan od lekova je da se iza zapete stavi jedan blanko, premda će to pomeriti znak za neki deo milimetra ulevo u odnosu na željenu poziciju u samom centru odgovarajuće kućice.

ISO I.B1.015

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p		Н	Č	░	└	╨	µ	н
1			!	1	A	Q	a	q	А	Њ	Ć	▒	┴	╤	а	њ
2			"	2	B	R	b	r	Б	О	Đ	▓	┬	╥	б	о
3			#	3	C	S	c	s	В	П	Š	│	├	╙	в	п
4			$	4	D	T	d	t	Г	Р	Ž	┤	─	╘	г	р
5			%	5	E	U	e	u	Д	С	α	╡	┼	╒	д	с
6			&	6	F	V	f	v	Ђ	Т	β	╢	╞	╓	ђ	т
7			'	7	G	W	g	w	Е	Ћ	Ω	╖	╟	╫	е	ћ
8			(	8	H	X	h	x	Ж	У	č	╕	╚	╪	ж	у
9			)	9	I	Y	i	y	З	Ф	ć	╣	╔	┘	з	ф
A			*		J	Z	j	z	И	Х	đ	║	╩	┌	и	х
B			+	;	K	[	k	{	Ј	Ц	š	╗	╦	█	ј	ц
C				<	L	\	¦	|	К	Ч	ž	╝	╠	▄	к	ч
D				=	M	]	m	}	Л	Џ	·	╜	═	▌	л	џ
E				>	N	^	n	~	Љ	Ш	«	╛	╬	▐	љ	ш
F			/	?	O	_	o		М	π	»	┐	╧	▀	м	

Računari 86

532 [BAZE CLIPPER, PC] Na slikama 2, 3 i 4 prikazani su zanimljivi i, verujemo, korisni potprogrami koji će obradovati svakoga ko se bavi programiranjem u *Clipper*-u na nešto nižem nivou (direktna komunikacija sa memorijom, hardverom itd). Prevedite SPEX50.ASM sa slike 2 primenom MASM-a ili nekog kompatibilnog asemblera (pri tome se automatski učitava CLIPPER.INC sa slike 3) a zatim iz programa koristite neku od funkcija AAD, AOR, AXOR, PEEK, POKE (značenje jasno svakome ko je programirao na bejziku, a ko od nas nije?), PUTCOLOR (promena atributa znakova u prozoru zadatih koordinata, pogledajte zaglavlje sa slike 4 koje određuje boje) i GETCLR (vraća vrednost kolor atributa *16*boja_pozadine + boja_prednjeg_plana*, zgodno za kombinovanje sa PUTCOLOR). Potprogram možete da povežete sa vašim programom komandom **RTLINK FILE ime,SPEX50** ili da ga ugradite u biblioteku programa koje stalno koristite. [Preuzeto sa Sezama, autor **Petar Stojanović** (*pstojanovic*); sa Sazama možete preuzeti izvorni kod ovog programa, kao i par primera upotrebe pisanih u *Clipper*-u]

533 [BASE CLIPPER, PC] Ako vam nekada zatreba da tekst iz nekog tekst procesora ubacite u DBF datoteku kao *memo* polje, snimite ga u *Generic* formatu tako da u njemu ostanu samo tabulatori i oznake novih redova. To je upravo format koji koristi i Clipper. U *WordPerfect*-u tekst se snima u ovom obliku sa *Text In/Out* (Ctrl F5) A G. [Preuzeto sa Sezama, autor **Nenad Baločanin** (*nbalocanin*)]

534 [TEHP C, PC] Iako se ne smatra baš "čistom" tehnikom programiranja, upisivanje statusa i drugih sličnih informacija u samu EXE datoteku koja se izvršava je vrlo zgodno za kratka uslužna programa. U primeru sa slike 5 struktura *save* će biti ugrađena u program – u nju možemo da upišemo sve podatke koje treba čuvati. U primeru se ta struktura sastoji od nekoliko članova od kojih je član *find[6]* obavezan jer se po njemu pretražuje izvršni kod, a *offset* ubrzava pronalaženje iste strukture pri sledećim pozivima. Ostali elementi strukture mogu se dodavati prema potrebama – ovde je to samo promenljiva *flags* koja čuva vrednosti parametara. [Preuzeto sa Sezama, autor **Miljan Jovanović** (*mjova*)]

535 [HARDW, PC] Poput svih računara koji su "prijateljski raspoloženi prema korisniku", i PC se trudi da sve eventualne greške i probleme propratí odgovarajućom porukom koja se ispisuje na ekranu. Nevolja je, međutim, što u pojedinim fazama, a naročito tokom inicijalizacije računara, po ekranu ne može da se piše – ako ne mikroprocesor "probudi" i ustanovi da ne može da komunicira sa video karticom, nema ništa od ispisivanja izveštaja o tome! No, poruke se mogu poslati i na druge načine, recimo zvukom. Autori raznih BIOS-a su predvideli kombinacije pisaka koji znače razne stvari; ovoga puta navodimo "šifarnik" kod nas najpopularnijeg AMI BIOS-a. Jedan pisak označava neuspelo osvežavanje memorije (*DRAM refresh failure*), dva piska problem se modulom za proveru parnosti (*Parity circuit failure*), tri nemogućnost adresiranja početka memorije (*Base 64k RAM failure*), četiri probleme sa oscilatorom (*System timer failure*), pet problem sa samim procesorom (*Procesor failure*), šest neispravan kontroler tastature (*Keyboard controller – Gate A20 error*), sedam tonova je problem sa virtualnim modom 80386 (*Virtual mode exception error*), osam je greška u video memoriji (*Display memory R/W test failure*) a devet pogrešan čeksum samog BIOS-a (*ROM BIOS checksum failure*). Sve greške osim pretposlednje su fatalne, odnosno moraju se otkloniti da bi se sistem normalno "podigao". [Preuzeto sa Sezama, autor **Dragan Obrenović** (*obren*)]

536 [OS WINDOWS 487, PC] Pre par meseci pisali smo o tome da se *Windows* može aktivirati sa SYSTEM\WIN386 umesto standardnim WIN – štedi se nešto memorije i izbegava prikazivanje naslovnog ekrana. Problem, međutim, nastaje kada iz *Control panel*-a pokušate da promenite veličinu *swap* fajla – pojeviće se pitanje želite li da restartujete *Windows*, a zatim ćete biti nemilosrdno izbačeni u DOS. Promena veličine *swap* fajla na ovaj način neće biti izvršena, i čak ćete po ulasku u *Windows* dobiti obaveštenje da *swap* fajla uopšte nema i da se mora raditi sa privremenim. Ukoliko, međutim, *Windows* startujete "uobičajenim" WIN, čitava operacija se obavlja na prirodan i "očekivani" način. Ukratko, aktiviranje *Windows*-a sa SYSTEM\WIN386 ne treba koristiti ako se menjaju parametri *swap* fajla ili možda i neki drugi sistemski parametri. [Preuzeto sa Sezama, autor **Drazen Pantić** (*drazen*)]

537 [OS WINDOWS, PC] Ako imate računar sa

Listing 2 SPEX50.ASM

```
;****************************************************************
; SPEX50 ASM
;
; Mala biblioteka pomoćnih rutina za Clipper
; BY Stojanović 1992
;
; Procedure:
;
; SLAN(Segment,Ofset,VREDNOST)
;     Upisuje VREDNOST u memoriji na Segment:Ofset
;
; PUTCOLOR(Y1,X1,Y2,X2,COLOR)
;     Menja kolor atribut na ekranu u regionu
;     zadatom koordinatama Y1,X1,Y2,X2
;
; Funkcije:
;
; PEEK(segment,ofset):(BYTE)
;     Vraća podatak sa zadane Segment:Ofset
;
; GETCLR(Foreground,Background):(BYTE)
;     Kombinuje boje teksta i pozadine u atribut
;     (računa Background*16+Foreground)
;
; SCRCHAR(Y,X):(BYTE)
;     Vraća ASCII kod karaktera na Y X
;
; AAND AOR AXOR:(VAR1 VAR2):WORD
;     Vraća rezultat operacija AND, OR, XOR
;
;****************************************************************
                NAME            SPEX50
include clipper.inc

MONOVMODE       equ             0B000h
COLORVMODE      equ             0B800h

DEFPARPUBLIC    <PEEK,POKE,PUTCOLOR,GETVMODE>
DEFPARPUBLIC    <GETCLR,SCRCHAR,AAND>
DEFPARPUBLIC    <AAOR,AOR,AXOR>
EXTERN

BEGINDATA
                SEGADDR         dw      ?
                OFSADDR         dw      1
                VALUE           db      1
                COLORPUT        db      1
                X1              db      1
                X2              db      1
                Y1              db      1
                Y2              db      1
                VIDEOADDR       dw      1
                X               db      ?
                Y               db      1
                FOREGROUND      db      ?
                BACKGROUND      db      ?
ENDDATA

BEGINCODE

                BEGINLOCALPROC  GETVADDR
                POPFROMADDR     0000h,0049h,VALUE
                cmp             VALUE,7
                jne             MONOVMODE
                mov             VIDEOADDR,COLORVMODE
                ret
MONOVMODELBL:
                mov             VIDEOADDR,MONOVMODE
ENDLOCALPROC

BEGINPROC       SCRCHAR
                GETI            <Y>
                GETI            <X>
                call            GETVADDR
                mov             al,y
                mov             bl,80
                mul             bl
                add             al,x
                shl             ax,1
                POPFROMADDR     VIDEOADDR,ax,VALUE
                PUTI            <VALUE>
ENDPROC

BEGINPROC       PUTCOLOR
                pushf
                GETI            <Y1>
                GETI            <X1>
                GETI            <Y2>
                GETI            <X2>
                GETI            <COLORPUT>
                call            GETVADDR
                cld
                push            VIDEOADDR
                pop             es
                AFFOR           Y,Y1,Y2,1
                mov             al,y
                mov             bl,80
                mul             bl
                xor             bx,bx
                mov             bl,x1
                add             ax,bx
                shl             ax,1
                mov             di,ax
                inc             di
                xor             cx,cx
                mov             al,COLORPUT
                mov             cx,cx
                mov             cl,x2
                sub             cl,x1
                inc             cx
LOOPI1:
                stosb
                inc             di
                loop            LOOPI1
                AFNEXT
                popf
ENDPROC

BEGINPROC       SLAN
                GETI            <SEGADDR>
                GETI            <OFSADDR>
                POPFROMADDR     SEGADDR,OFSADDR,VALUE
                PUTI            <VALUE>
ENDPROC

BEGINPROC       POKE
                GETI            <SEGADDR>
                GETI            <OFSADDR>
                GETI            <VALUE>
                INT             1
                PUTONADDR       SEGADDR,OFSADDR,VALUE
ENDPROC

BEGINPROC       AAND
                GETI            <SEGADDR>
                GETI            <OFSADDR>
                mov             ax,SEGADDR
                add             ax,OFSADDR
                mov             VIDEOADDR,ax
                PUTI            <VIDEOADDR>
ENDPROC
BEGINPROC       AOR
                GETI            <SEGADDR>
                GETI            <OFSADDR>
                mov             ax,SEGADDR
                or              ax,OFSADDR
                mov             VIDEOADDR,ax
                PUTI            <VIDEOADDR>
ENDPROC
BEGINPROC       AXOR
                GETI            <SEGADDR>
                GETI            <OFSADDR>
                mov             ax,SEGADDR
                xor             ax,OFSADDR
                mov             VIDEOADDR,ax
                PUTI            <VIDEOADDR>
ENDPROC

BEGINPROC       GETCLR
                GETI            <FOREGROUND>
                GETI            <BACKGROUND>
                xor             ax,ax
                mov             al,BACKGROUND
                shl             ax,4
                add             al,FOREGROUND
                push            ax
                call            __retni
                add             sp,2
ENDPROC
ENDCODE
```

Mala biblioteka pomoćnih rutina za *Clipper*

Listing 3

```
LIST
286c
;****************************************************************
; clipper.inc
;
; BY by Petar Stojanović 1992
;
;****************************************************************

$force=7

$ITEM           MACRO           a1
                EXTRN           __RETNI:FAR
                EXTRN           __PARNI:FAR
                EXTRN           __PARINFO:FAR
                IRP             $ITEM,<P1>
                EXTRN           $ITEM:FAR
                ENDP
ENDM

DEFPARPUBLIC    MACRO           P1
                IRP             $ITEM,<P1>
                PUBLIC          $ITEM
                ENDM
ENDM

BEGINDATA       MACRO           P1
                DGROUP          group datanamer
                DATANAME        segment public 'DATA'
BEGINCODE       MACRO P1
                codenamer       segment 'CODE'
                assume          cs:codename ds:DGROUP
ENDDATA DEBUG
codename        ELSE
                end
                ENDM
                ENDM
ENDDATA         MACRO
                datanamer       ends
                ENDM
ENDM

GETI            MACRO           p1
                mov             ax,$PNL
                push            ax
                $PNL=$PNL+1
                call            __parni
                add             sp,2
                IF              TYPE P1 EQ 1
                mov             p1,al
                ELSE
                mov             p1,ax
                ENDIF
ENDM

PCOUNT          MACRO           P1
                mov             ax,0h
                push            ax
                call            __parinfo
                add             sp,2
                IF              TYPE P1 EQ 1
                mov             P1,al
                ELSE
                mov             P1,ax
                ENDIF
ENDM

BEGINPROC       MACRO           PROCNAME
$PNL=1
PROCNAME        proc            far
                push            bp
                mov             bp,sp
                push            ds
                push            es
                push            si
                push            di
                push            ax
                push            bx
                push            cx
                push            dx
ENDPROC         MACRO
                pop             dx
                pop             cx
                pop             bx
                pop             ax
                pop             di
                pop             si
                pop             es
                pop             ds
                pop             bp
                retf
PROCNAME        endp
                ENDM
ENDM

PUTONADDR       MACRO           SEGMENTA OFSETA,VALUEJA
                push            SEGMENTA
                pop             es
                mov             di,OFSETA
                IF              TYPE VALUEJA EQ 1
                mov             ax,VALUEJA
                mov             word ptr es:[di],ax
                ELSE
                mov             al,VALUEJA
                mov             byte ptr es:[di],al
                ELSEIF
ENDM

POPFROMADDR     MACRO           SEGMENTA OFFSETA,VALUEJA
                push            SEGMENTA
                pop             es
                mov             di,OFSETA
                IF              TYPE VALUEJA EQ 2
                mov             ax,word ptr es:[di]
                mov             VALUEJA,ax
                ELSE
                mov             al,byte ptr es:[di]
                mov             VALUEJA,al
                ENDIF
ENDM

PUTI            MACRO           P1
                IF              TYPE P1 EQ 1
                xor             ax,ax
                mov             al,p1
                ELSE
                mov             ax,p1
                ENDIF
                push            ax
                call            __retni
                add             sp,2
ENDM

BEGINLOCALPROC  MACRO           P1
P1              proc            near
ENDLOCALPROC    MACRO
                ret
P1              endp
ENDP
ENDM

AFFOR           MACRO           VAR,FROMVAR,TOVAR,STEPVAR
                LOCAL           FORCE db $force
                push            ax
                IF              TYPE FROMVAR EQ 1
                mov             ax,ax
                mov             al,FROMVAR
                ELSE
                mov             ax,FROMVAR
                ENDIF
                IF              TYPE VAR EQ 1
                mov             VAR,al
                ELSE
                mov             VAR,ax
                ENDIF
FLAFORCEA:
                pop ax

AFNEXT          MACRO
                push            ax
                IF              TYPE VAR EQ 1
                mov             ax,ax
                mov             al,VAR
                ELSE
                mov             ax,VAR
                ELSEIF
                add             ax,STEPVAR
                IF              TYPE VAR EQ 1
                mov             VAR,al
                ELSE
                mov             VAR,ax
                ENDIF
                IF              TYPE TOVAR EQ 1
                xor             ax,ax
                mov             al,TOVAR
                ELSE
                mov             ax,TOVAR
                ENDIF
                IF              TYPE TOVAR EQ 1
                cmp             VAR,al
                ELSE
                cmp             VAR,ax
                ENDIF
                jle             FLAFORCE
                pop ax
                $force=$force+1
                ENDM
ENDM
```

Makro definicije

Listing 4

```
#define BLACK          0
#define BLUE           1
#define GREEN          2
#define CYAN           3
#define RED            4
#define MAGENTA        5
#define YELLOW         6
#define WHITE          7
#define BRIGHT         8
#define BLINK        128

#define DIV(Arg1,Arg2)          (Round(Arg1/Arg2,0))
#define FOREGROUND(Color)       (Mod(Color,16))
#define BACKGROUND(Color)       (DIV(Color,16))
```

Zaglavlje sa definicijama boja

SCSI diskom, možda ćete morati da koristite /DOUBLE_BUFFERING opciju keš programa SMARTDRV. Ovo dvostruko baferisanje obezbeđuje Windows-u rad sa SCSI adapterima koji ne "umeju" da rade sa virtuelnom Windows memorijom.

SMARTDRV linija sa opcijom /DOUBLE_BUFFERING se postavlja u CONFIG.SYS, u AUTOEXEC.BAT ostaje SMARTDRV linija sa ostalim parametrima. SETUP će sve ovo uraditi automatski – ako ne može da "odluči" da li je dvostruko baferisanje na datom sistemu neophodno ili nije, ugradiće ga "za svaki slučaj".

Kako da odredite da li je dvostruko baferisanje zbilja potrebno? Ugradite odgovarajuću liniju u CONFIG.SYS, resetujte računar, pokrenite *Windows 3.1*, aktivirajte DOS sesiju, tamo otkucajte SMARTDRV i ako u svakom redu tabele u koloni *Buffering* vidite NO, možete mirne duše ostati pri "normalnom" baferisanju.

Sve ovo je teorija, u praksi, stvar ne funkcioniše baš ovako sa *Adaptec AHA-1542B SCSI* adapterom. Sa ili bez dvostrukog baferisanja, izveštaj u SMARTDRV u kaže da baferisanja nema. Naravno, treba reći da je instaliran *Adaptec ASPI4DOS* drajver za ovaj adapter, koji služi da ga prilagodi i optimizuje korišćenje pod DOS-om i *Windows*-om. Takođe, OEMM 386 je instaliran na sistemu sa opcijom DISKBUF=6. Parametar "6" kaže da se za *bus-master adapter* ostavlja 6 K za pristup disku, odnosno uređajima na SCSI magistrali.

Na kraju, ako je na računaru instaliran OEMM, ne sme se dozvoliti OPTIMIZE-u da "podigne" SMARTDRV /DOUBLE_BUFFERING u visoku memoriju. Da bi dvostruko baferisanje stvarno radilo, SMARTDRV se mora nalaziti u osnovnoj memoriji. [Preuzeto sa Sezama, autor **Zoran Kehler** (*zkehler*)]

538 [COMM] Da li ste se nekada zapitali koliko se brzina prenosa može "izvući" iz nekog 2400 bps modema? Pokušaćemo da odgovorimo na to pitanje kroz malo teorije. Uzmimo za početak običan modem, dakle bez MNP-a. Njegova brzina prenosa je 2400 bps i to kako na relaciji modem-računar, tako i na relaciji modem-drugi_modem. Prenos je asinhroni, najčešće u formatu 8 bita podataka, jedan start i jedan stop bit, što znači da jedan karakter (znak) čini 10 bita. Ako pretpostavimo da između dva bajta ne postoji posebne pauze, protok bi bio oko 2400 bita u sekundu / 10 bita po bajtu = 240 cps (znakova u sekundu). Posmatrajmo sada prenos datoteka. Na svakih 1024 bajta (prenos *ZModem*-om uz čistu vezu) oba računara računaju kontrolnu sumu (CRC) tog bloka, jedan šalje taj CRC, drugi upoređuje pa šalje sekvence za prijem novog bloka od 1024 (ili manje) bajta, itd. Ukoliko za svaki blok se razmenjuju neke kontrolne sekvence, što rezultuje određenim usporenjem, tipično na 232-234 cps. Noviji *ZModem* protokoli optimizuju komunikaciju pa je protok nešto veći, ali i dalje **ispod** teorijskih 240 cps.

Prelazimo sada na MNP 4. Ovde je situacija malo složenija, pošto je veza modem-računar asinhrona a veza modem-drugi_modem sinhrona, dakle **bez** start i stop bitova. Prenosi se, dakle, 8 bita po bajtu, što znači da je maksimalna teorijska brzina prenosa 2400 bps/8 bpB=300 cps, znatno više od protoka na relaciji modem-računar, što znači da brzinu te komunikacije treba povećati (već smo pisali da MNP modeme, bez obzira na deklarisanu brzinu od 2400 bps, u komunikacionom programu treba podesiti da rade na 9600 bps). S obzirom da MNP 4 podrazumeva kontrolu grešaka koja se obavlja u realnom vremenu (slično kao kod prenosa datoteka), stvarna brzina je nešto manja. Pri prenosu datoteka važe pravila sasvim slična opisanima, pa se za *ZModem* dobija oko 276 cps, odnosno, kod novijih verzija, oko 280 cps. U ovom slučaju se očito vrši dvostruka kontrola grešaka pri prenosu datoteka – jedna je na nivou same MNP 4 komunikacije a druga na nivou *ZModem* protokola. Da bi se to dupliranje izbeglo, izmišljen je protokol *Ymodem G* kod koga se CRC računa i razmenjuje samo na kraju celog transfera, što bi trebalo da rezultuje protokom koji je sasvim približan maksimalno mogućem. Zvuči lepo, ali može da donese veliki problem, ako jedan od modema permanentno šalje podatke ali njihov prenos ne uspeva, bafer modema će se najzad napuniti i neki podaci će možda biti izgubljeni. Ako se ovo dogodi u toku *Ymodem G* prenosa, jedino što se može učiniti je da se čitav prenos ponovi, naravno bez garancije da se i u drugom pokušaju neće dogoditi isto. Zato preporučujemo upotrebu mnogo robusnijeg *ZModem-a*: osam ili manje procenata ušteda koju *Ymodem-G* nudi teško da su vredni ovoga rizika.

MNP 5 obuhvata sve što je rečeno za MNP 4 i kao dodatak obezbeđuje *on-line* kompresiju podataka. Kako se to odražava na prenos datoteka? Ako se vrši prenos već arhiviranog (npr. primenom PKZIP-a ili ARJ-a) fajla, odlazi do usporenja u odnosu na MNP 4 pošto modem (bezuspešno) pokušava da komprimuje blokove, za šta mu je potrebno određeno vreme. Kod nearhiviranih fajlova dobitak u brzini može da bude spektakularan i da ide do 4:1. U praksi se za tekstualne datoteke može postići otprilika 650 cps.

Čitava ova diskusija se odnosi na **hardverski** MNP. Neki komunikacioni programi obezbeđuju MNP emulaciju na ne-MNP modemima, ali s obzirom da takvi modemi nemaju mogućnost sinhrone komunikacije, na njima se ne može postići ništa više od onih bez MNP-a maksimalnih 240 cps. Ovaj broj je dalje umanjen vremenom potrebnim za softversku emulaciju MNP protokola.

Ostaje još da prokomentarišemo tvrdnje tipa "ovaj komunikacioni program obezbeđuje 300 cps ili više". Obično se naprosto radi o pogrešnom merenju – to može biti neka nepreciznost pri radu sa PC-jevim časovnikom realnog vremena ali i namerno laganje u reklamne svrhe. Zato svi koji dobiju "tolki" cps treba da pokušaju da nezavisnom štopericom izmere trajanje prenosa, podele dužinu datoteke tim vremenom i tako dobijenu vrednost uporede sa onom koju je pokazao komunikacioni program. U principu je za očekivati da eksterni protokoli koji su implementacija originalnog *Chuck Forsberg*-ovog sorsa (DSZ, GSZ itd) pokazuju tačne vrednosti, dok internim protokolima koji su ugrađeni u razne komunikacione programe ne treba previše verovati. [Preuzeto sa Sezama, autor **Bojan Stanojević** (*bojs*)]

539 [OS DOS, PC] Vlasnici nekih VGA karti priključenih na monohrom VGA monitor imaju probleme sa nijansama – u startu ih nema, a onda se u nekom trenutku ili posle startovanja nekog programa "iz čista mira" pojave. Rešenje je ugradnja komande MODE CO80 u AUTOEXEC.BAT – trebalo bi da problem nestane "kao rukom odnesen". Sličan efekat proizvodi i korišćenje drajvera ANSI.SYS (ili VANSI.SYS) a i startovanje igre *Test Drive II*. [Preuzeto sa Sezama, prilozi **Dragan Obrenović** (*obren*) i **Viktor Velisavljev** (*vvelisavljev*)]

540 [KOMS, PC] Ako pri instalaciji nekog novijeg *Borland*-ovog paketa dobijete poruku *Problem with unzip.exe C)ontinue or A)bort*, stvar nije tako stra-

Listing 5

```
struct {
    char find[8];
    unsigned long offset;
    byte flags;
} save= {
    "STAMP"
    SW_BYTES | SW_TOTAL
    0L
};

void cfg_save(char *exename) {
    FILE *exe;
    char buffer[sizeof(save.find)-1];
    word c;

    exe= fopen(exename "rb+");
    if(!exe) {
        fprintf(stderr "Error: cannot open file. CFG not saved!\n");
        exit(-1);
    }

    if(!save.offset) {
        c= fgetc(exe);
        while(!feof(exe)) {
            if(c == (word)save.find[0]) {
                fread(buffer sizeof(buffer) 1 exe);
                if(!memcmp(buffer &save.find[1], sizeof(buffer))) {
                    save.offset= ftell(exe) sizeof(buffer) 1;
                    break;
                }
                fseek(exe -sizeof(buffer) 1 SEEK_CUR);
            }
            c= fgetc(exe);
        }
    }
    if(feof(exe)) {
        fprintf(stderr "Error: cannot find marker CFG not saved!\n");
        exit(-2);
    }
    fseek(exe save.offset SEEK_SET);
    fwrite(&save sizeof(save) 1 exe);
}

void help(void) {
    printf("Usage: ex [/t] [/k] [/w] [drive] \n" file);
    printf("       /t - only TOTAL line\n");
    printf("       /k - memory status in kb\n");
    printf("       /s - save options\n");
    printf("       [drive] - drive letter(s)\n");
}

void main(int argc char *argv[]) {
    int i= 1
        savecfg= 0;

    while(argc > i) {
        switch(argv[i][0]) {
          case '/':
          case '-':
            switch(tolower(argv[i][1])) {
              case 'h':
              case '?':
                help();
                exit(1);
              case 'k':
                save.flags^= SW_BYTES;
                break;
              case 't':
                save.flags^= SW_TOTAL;
                break;
              case 's':
                savecfg= 1;
                break;
            }
            break;
          case '?':
            help();
            exit(1);
            break;
        }
        i++;
    }

    if(savecfg) {
        cfg_save(argv[0]);
        printf("\nCFG saved.\n");
        exit(0);
    }

    /* iavrsi dan programa ... */
```

Upisivanje statusa u .EXE datoteku

Listing 6

Escape **sekvence za VT terminale**

šno kao što izgleda. Instalacioni program radi tako što na disk prenese prvo samoga sebe, zatim READ.COM i najzad UNZIP.EXE, a ovaj kreće sa raspakivanjem. Ako ste na isnatalacionoj disketi greškom uradili DS EN ili nešto slično tome, INSTALL će (ne vodeći računa o imenu... možda je to neka mala zaštita) prekopirati prva tri programa koja nađe a onda pokušati da startuje poslednji od njih. Ako to nije UNZIP.EXE nego, na primer, neka arhiva, ostaje samo da se prijavi greška. [Preuzeto sa Sezama, autor **Zoran Brkić** (*zoob*)]

541 [COMM]: Na slici 6 date su *escape* sekvence karakteristične za VT terminale priključene na veći nu domaćih VAX-ova. Gotovo svi komunikacioni programi podržavaju VT100 emulaciju pa, prema tome, i većinu nabrojanih sekvenci; ukoliko, međutim, poželite da nešto sami isprogramirate, tabela će vam biti dragocena. [Preuzeto sa *Pegasus BBS*-a, Švajcarska]

542 [IGRE; PC]: Za ljubitelje igre GODS, šifre koje će vas dovesti na više nivoe su ZRK, DUE i PPO. [Preuzeto sa Sezama, autor **Atila Cigan** (*bearboy*)]

Događej avgusta na Sezamu svakako je uvođenje prve višekorisničke igre, koje je izazvala veliko interesovanje i izazvala veliki broj poruka u konferenciji SEZAM i u ostalim konferencijama je, i pored sezone godišnjih odmora, bilo veoma „živo", pri čemu je najviše poruka stiglo u „političku" konferenciju FORUM. No, da pođemo redom.

SEZAM

SISTEM ZA MODEMSKE KOMUNIKACIJE

15

Priprema: Dejan Ristanović u saradnji sa moderatorima konferencija

Moderatori: Dejan Batinda (CIVILIZACIJA), Nikola Bošković (PC USER), Jovan Bulajić (FILES), Aleksandar Damnjanović (PC OS), Slobodan Kalezić (ORKA), Vladimir Kostić (muzika), Vladimir Kraljević (VICEVI), Esad Jakupović (RAČUNARI), Dalibor Lenik (ATARI ST), Bojan Pavković (PC UTIL), Pavle Peković (PC PROG), Dejan Ristanović (NOVOSTI i KOMUNIKACIJE), Bojan Tešović (FORUM), Dejan Veselinović (PC HARD) i Zoran Životić (SEZAM)

Tokom avgusta smo „opslužili" 21 372 poziva, komunicirali sa korisnicima 4 837 časova (tj. gotovo 202 dana (s obzirom da avgust ima 31 dan, u proseku je 6,5 nodova bilo **neprekidno** zauzeto), od čega je u *chat* u provedeno 915 časova ili 114 radnih dana. Korisnici su se Sezama *download*-ovali 874 megabajta podataka, dok je za isto vreme sistemu poslalo oko 40 megabajta. Najviše poziva uputili su (ne računajući službena lica Sezama) **Milica Nešović** (*mnesovic*), **Stanislav Šokorac** (*ssokorac*) i **Ivan Simanković** (*ivans*), najviše vremena na vezi proveli su **Vesna Denčić** (*veca*), **Miloš Prvulović** (*prvul*) i **Bojan Pavković** (*bojanp*), dok su najviše *download*-ovali **Slobodan Petrović** (*feniks*), **Bora Živanović** (*bora*) i **Andor Gindei** (*pishta*).

Što se „trajnih vrednosti" tiče, tokom avgusta je u javne konferencije pristiglo oko 7 400 poruka (gotovo 6,2 megabajta teksta) i 2,4 megabajta datoteka priključenih uz njih, 6 618 privatnih poruka (preko 2,5 megabajta privatne pošte) i 1 867 poruka u grupama (664 kilobajta teksta, ne računajući datoteke priključene uz te poruke).

Prva višekorisnička igra na Sezamu zove se HANGMAN – u pitanju je zapravo jednostavna „pilot igra" koja treba da ispita interesovanje korisnika za takve aktivnosti (ako ste poslednjih dana pokušavali da dobijete Sezam, svakako znate da je to interesovanje popriličnо). Pravila su sasvim jednostavna – slovo po slovo, igrač pokušava da pogodi zadatu rečenicu. Ako pogodi slovo, ono se ispisuje na svim mestima na kojima je ranije bilo „pokriveno" crticama. Ako, sa druge strane, otkuca slovo koje se u rečenici ne javlja, dobija izveštaj o tome i pokušaj se računa kao „greška" – posle nekoliko grešaka sistem proglašava da rečenica nije pogođena i igrač gubi bodove. Uspešno pogođena rečenica daje igraču pravo da zada novu rečenicu koju će ostali igrači pogađati. Igrač ne dobija bodove samo kada pogodi rečenicu (tada broj bodova zavisi od broja načinjenih grešaka); ukoliko neko drugi ne uspe da pogodi rečenicu, određeni bodovi se pripisuju njenom autoru. Posle fiksnog roka partija se završava, a igrač koji je sakupio najviše bodova biva proglašen za pobednika – pobednik je ujedno i moderator sledeće partije HANG-a i ima mogućnost da zada tematiku kojom će se partija baviti, broj poena koje igrač dobija i gubi u raznim situacijama, maksimalni broj rečenica u igri i tome slično. U avgustu je odigrano nekoliko partija HANG-a – prve dve su bile pristupačne samo moderatorima dok je program testiran, prva „javna" partija je imala „slobodnu temu", a sledeće su zahtevale da se unose naslovi filmova (moderator **Miloš Prvulović** (*prvul*)), pesama (moderator **Aleksandra Petrović** (*acka*)) i knjiga (moderator **Vesna Denčić** (*veca*)). Posebnu pažnju izazvala je jedna probna partija HANG-a koja se, na zahtev korisnika, „produžila" kroz čitav avgust – trebalo je pogađati i unositi rečenice o Anni (*anna* je **Ana Dimitrijević**, naša korisnica) – ako vas zanima 1500 rečenica o Ani, svakako pozovite Sezam! Ukratko, bilo je veselo. Program HANG je u potpunosti integrisan u Sezam, a njegov autor je **Dejan Ristanović**.

Kada se na sistemu pojavi neka novitet, konferencija **SEZAM** je uvek puna diskusija o njemu – tako je i novouvedeni HANG doneo preko 500 poruka u temi *igre* (višekorisničke igre). Osim pohvala, bilo je i mnogo različitih ideja, koje su se uglavnom bavile problemima sa slovima čćđšž – jedna od specifičnosti Sezama je da korisnik može koristiti (gotovo) proizvoljan raspored YU slova (većina i dalje koristi YUASCII ali su i drugi rasporedi popularni), dok se sistem brine o konverzijama, ali korisnici koji nisu uopšte imali ugrađena YU slova (SET CODE NONE na Sezamu) nisu mogli da igraju. To je rešeno specijalnim modom za kucanje YU slova kao dvoznaka (npr. %cc je ć) koje program HANG docnije konvertuje u „normalna" slova. Novim standardnim rasporedom YU slova koji predlaže Savezna komisija bavila se tema *yu kod*, dok smo u ostalim temama uglavnom pokušavali da pomognemo početnicima u „snalaženju" na Sezamu.

S obzirom da naš list u avgustu ne izlazi, u konferenciji **RAČUNARI** je bilo sasvim malo poruka. Tek pred kraj meseca, najava septembarskih „Računara" donosi prve komentare, koji su se naročito odnosili na vest o novoj kompresionoj tehnici („Računari 84", strana 4), koju je velika većina diskutanata procenjuje da nije moguća. Za razliku od prethodne „nezamislive" cene od 8000 (tadašnjih) dinara, nova cena „Računara" prilično je povoljno primljena.

U „modemskim" temama konferencije **KOMUNIKACIJE** bilo je reči o 9600 bps i bržim modemima i njihovoj primenljivosti u našim uslovima, pa i na Sezamu (pokazuje se, na žalost, da takvih modema kod nas i dalje ima jako malo), ZOOM S/R fax modemu i njegovom podešavanju za rad pod programom *BitFax*, te o razlikama između MNP4 i MNP5 (jedna od zbilja većih tema na Sezamu). Tema *kom programi* uglavnom se bavila novom verzijom među Sezamovcima najpopularnijeg komunikacionog programa (*Telemate 3.10*). Ona je donela relativno malo novih stvari, među kojima su najznačajniji prateći *IW* koji olakšava rad pod *Windows*-om, „inteligentno" citiranje sa preformatiranjem teksta i nestanak TMCFG promenljive. Bilo je zatim reči o konverziji BFX (*BitFax*) u TIF format, a veliki broj poruka se svodi i na Jovačke priče" o (ne)postignutim brzinama prenosa na pojedinim relacijama – pominju se čak i „neverovatne" cifre od 300 bps (bita u sekundu). Teme koje se odnose na elektronsku poštu i mini računare VAX su i dalje pune poruka koje se bave mogućnošću slanja e-*mail*-a po svetu posle „odsecanja" *Bitnet*-a kao i stranim BBS-ovima koji obezbeđuju takve usluge. Takva diskusija, naravno, teško prolazi bez pominjanja kvota i ograničenja na računarima beograd-skog Elektrotehničkog fakulteta (koji su pristupačni preko modema), ali su se ovoga puta na Sezam javili i programeri i operateri Računskog centra ETF da odgovore na najčešća pitanja. Tema *strani bbs* bavila se uglavnom BBS-ovima u susednim zemljama (Mađarska, Grčka itd.) kao i cenama poziva na ovim rastojanjima, u temi *yu bbs* opisivan je modemski pristup domaćim bibliotekama, dok je u temi *razno* kratko rečeno, bilo svega i svačega.

U konferenciji **PC HARD** diskutovalo se o prednostima matičnih ploča sa većim procesorskim kešom, o uključivanju i isključivanju internog keša mikroprocesora 80486, o prenošenju BIOS-a sa jednog računara na drugi, o izvršavanju programa *Mathematica 386* na DTK pločama, o džamperima na *Conner* diskovima, o diskovima bez auto parkiranja (zar to još postoji?), o vezi između dimenzija diska i dimenzije klastera (mnogi kažu da je logika koju primenjuje *Microsoft* totalno neopeka), o (domaćem) programu TESTIDE koji određuje tačne parametre IDE diska, o rasporedu mikroprekidača na *Trident 8800* VGA kartici, o sivim nijansama na monohrom monitoru (nije tako jednostavno kao što izgleda!), o programiranju video kontrolera *Hercules* kartice, mogućnostima rotiranja (*mirror*) ispisa na *LaserJet* štampačima (nerešivo jednom komandom, ali se može pisati program koji „invertuje" ispis), popravci originalnog IBM monitora i mnogim drugim temama. „Fil juž" diskusije je, kao i obično, bio *Stacker* koji je, nekada opravdano a često neopravdano, „okrivljen" za brojne havarije podataka na diskovima i razna druga probleme.

U konferenciji **PC OS** bilo je, kao i uvek, dosta reči o PC-jevim operativnim sistemima. U okviru tema *unix* bavili smo se upotrebom modema pod SCO operativnim sistemima. Dobili smo vest o novoj (32-bitnoj) verziji *Coherent*-a. Tema *windows* je bila ubedljivo najaktivnija: govorilo se o *TrueType* fontovima, promeni fonta iz kontrolnog panela DOS prozore, upotrebi *Windows 3.1* na SCSI diskovima, rasporedu kodova za YU slova, problemima sa poslednjom verzijom programa 800, pozivanju *Windows*-a direktno preko programa WIN386 (zaobilaženjem programa WIN.COM). Dobili smo programe *BigDesk*, *BackMenu* (rad sa više virtuelnih ekrana) i *WinPost* (za organizovanje podsetnika). Tema *4dos* je bila „zagušena" diskusijom o raznim programima koji obezbeđuju izbor različitih softverskih konfiguracija sistema pri podizanju (BOOT.SYS, BOOTCON itd.). U temi *dv* bilo je reči o problemima koje *DesqView* ima pri radu sa diskom. Tradicionalno najposećenija tema *ms dos* je ovoga meseca primila nešto manje poruka – znači li to da polako prelazimo na *Windows* i OS/2?

Učestvujući u diskusiji o bojama i organizaciji video memorije PC-ja (konferencija **PC PROG**), setili smo se i starih dobrih vremena *Spectrum*-a, QL-a i ostalih. Kroz konkretne primere korišćenja upoznali smo program OBJ2ASM (*Object TO Assembler*). Mišljenje da su viši programski jezici zadali konačan udarac asembleru je uzdrmano obiljem poruka u kojima se traži literatura i pomoć za prve „mašinske korake". „Večita" diskusija o YU slovima se prenela i u PC PROG tako što je **Đorđe Rainović** (*djrainovic*) ostavio svoj program „H" koji direktno u video memoriji vrši konverzije kodnih rasporeda. Pričalo se još o jednoj večitoj temi, rezidentnim programima (TSR) kao i o načinu da se programski prepozna tip procesora. Tema *pascal* je dobila dosta novih poruka u okviru kojih su rešavani problemi poziva eksternih programa i nedostatka memorije, pisane brze rutine za kopiranje datoteka, učitavani veliki fajlovi, procenjivana razlika između statičkih i dinamičkih struktura, otvarane datoteke za čitanje i pisanje, editovani fontovi... Još jedan napad korisnika *FoxPro*-a na *Clipper* je propao, niko nije promenio tabor. Rešeni su problemi sa bibliotekom CLIPINT (poziv BIOS-a iz *Clipper* programa). Diskutovalo se o brzini pretraživanja baza po-

SPECIJALNE USLUGE

PODSISTEM TANJUG

Podsistem „Tanjug" omogućava pristup profesionalnim servisima novinske agencije TANJUG. Radi se o ažurnim političkim, ekonomskim, društvenim, naučnim, sportskim i drugim vestima iz zemlje i sveta. Servisi neprekidno pristižu na Sezam preko stalno iznajmljene linije, a mogu ih korisnici prema potrebi čitati ili prenositi na svoje računare radi dalje obrade. Pristup ovom podsistemu ostvaruje se iz menija (MENU TANJUG) ili komandnog moda (naredba NEWS).

Pristup podsistemu „Tanjug" je opciona usluga Sezama i plaća se posebno. Za detalje oko pretplate i uslova korišćenja obratite se Novinskoj agenciji Tanjug. Kontakt: **Milena Sekulić**, tel. 625-722.

PODSISTEM BETA

Podsistem „Beta" obezbeđuje praćenje domaćih časopisa u elektronskoj formi. Časopisi pristižu na Sezam pre nego u trenutku izlaska na kioske i dostupni su prema potrebama korisnika mogu pregledati, čitati ili prenositi na sopstvene računare. Pristup ovom podsistemu ostvaruje se iz menija (MENU BETA) ili komandnog moda (naredba BETA). Trenutno se na ovaj način distribuira nedeljnik „Vreme", a pristup je u probnom periodu slobodan.

Pristup podsistemu „Beta/Press" je opciona usluga Sezama i plaća se posebno. Za detalje oko pretplate i uslova korišćenja obratite se Marketingu lista „Vreme". Kontakt: **Vojislav Milošević**, tel. 646-070.

PODSISTEM BERZA

Berza je „Banka poslovnih podataka privrede Srbije" – radi se o osnovnim podacima o radnim organizacijama u Srbiji i njihovoj delatnosti, zajedno sa nazivima artikala koje pojedine firme traže ili nude. Različitim metodama pretraživanja mogu se utvrditi firme koje traže ili nude ono što vam je potrebno, a zatim proučiti ostali podaci o njima, koji uključuju adresu, telefon, teleks i faks. Imate mogućnost da na sličan način date podatke o ponudi i tražnji vaše firme, koje će drugi korisnici docnije pregledati. Pristup ovom podsistemu ostvaruje se iz menija (MENU BERZA) ili komandnog moda (naredba STOCK).

Pristup podsistemu BERZA je opciona usluga Sezama i plaća se posebno. Za detalje oko pretplate i uslova korišćenja obratite se firmi Electro Engineering, tel. 602-453.

tatska praćeno „takmičenja" u pisanju što kraće procedure za prebacivanje iz *insert* u *overtype* mod i obilno izložene razlika između *#include* i *do* i sve prednosti programa MAKE kao i prednosti korišćenja PLL fajlova za ubrzano linkovanje. Još jednom se potvrdilo da *Clipper* u 5.01 najviše odgovara RTLINK. Izgleda da je avgust pravi mesec da se krene sa programiranjem na C-u, a na takav zaključak smo došli čitajući brdo poruka u kojima početnici traže savet za odgovarajuću knjigu, kompajler, prve korake... Videli smo na koliko se načina može postaviti pointer na neku *far* adresu, upoznali se sa dinamičkim linkovanjem, formatiranjem ispisa brojeva, čitanjem znaka koji se nalazi na tekućoj poziciji kursora, preusmeravanjem izlaza, prevođenjem brojeva između različitih brojnih sistema.

Udarana tema konferencije **PC USER** bila je pomalo neočekivano, ona sa najopštijim imenom – *razno*. Razgovaralo se, međutim, o vrlo konkretnoj stvari – programima za prikazivanje dimenzija DOS particija i slobodnog prostora na njima. Tri korisnika Sezama, **Predrag Todorović, Robert Popović, Miljan Jovanović** (*todorp, robert* i *mjova*) su napisala programe tog tipa, zatim ih uz pomoć ostalih korisnika testirali i najzad stavili u javno vlasništvo. Testiranja tih korisnih uslužnih programa pratio je respirava o radu samog DOS-a i o problemima prevodilaca koji se koriste. S obzirom da je sve teklo „interaktivno", programi su prilagođeni željama i potrebama korisnika koji su učestvovali u diskusiji; programi su, osim toga, po svojim mogućnostima isti ili bolji od stranih programa tog tipa, što je još jedan dokaz da domaćim programerima ne manjka ni znanje ni sposobnost. Tema *grafika* se uglavnom bavila rešavanjem problema koji se javljaju pri unošenju geografskih karata u računar. Ispostavilo se da je jedno od mogućih rešenja *AutoCAD*, s tim što je u većini slučajeva za dalji rad s takvom bazom neophodno napisati potrebne ekstenzije u *Lisp*-u. O *AutoCAD*-u se diskutovalo i u temi *cad* – razmatrana je mogućnost ubrzavanja programa korišćenjem matematičkih koprocesora koji podržavaju hardversko množenje matrica. Konstatovano je, osim toga, da HIMEM.SYS više odmaže nego što pomaže *AutoCAD*-u, dok je QEMM po tom pitanju neutralan. Što se tekst procesora tiče, najviše smo diskutovali o *WordPerfect*-u i uključivanju činilice u tekstove koji se obrađuju. Tema *memory.mng* se, za razliku od ranijih meseci, bavila programima za upravljanje memorijom na AT (80286) računarima – izgleda da vlasnici 386 računara više nemaju ni problema ni dilema o korišćenju memorije! Pričalo se i o *Borland*-ovom paketu *Paradox Engine*, o instaliranju naših slova na matričnim štampačima kako hardverski tako i pomoću programa *Letrix*, o *Autodesk Animator*-u, za *Windows*, *PostScript* interpreterima, novim verzijama programa Shaz 8.0, PkLite 1.15, Scan 8.7B94, Clean 8.7B95, makroima u programskom editoru *Brief*.

Događaj meseca u konferenciji **ATARI ST** je, bez ikakve sumnje, program „StOR" koga je napisao **Dejan Anđelković** (*angel*). Reč je, kao što se vidi i iz imena, o ST verziji poznatog SOR-a, *Sezam Off-line Reader*-a. Iako još uvek u radnoj verziji i sa dosta detalja koje bi trebalo doraditi, ovaj program pokazuje da ima Atarista spremnih da se prihvate većeg posla. Osim toga, izgleda da i vrućina utiče na produktivnost programera: ovoga meseca dobili smo još nekoliko zgodnih i korisnih uslužnih programa. Pomenimo samo „Filer" koji je ST verzija programa za *Macintosh* koji obezbeđuje „cepanje" i „nadovezivanje" datoteka sa/na više delova; napisao ga je **Milan Adamov** (*madamov*). Tvrdnju da su najglasniji bili programeri potvrđuje i činjenica da su tema *programiranje* i *c* primile najviše poruka.

I ovoga meseca su igre privlačile najviše pažnje u konferenciji **ORKA**. Pored uobičajene razmene rešenja, saveta, šifara itd., ovoga meseca su slavu zaslužili neki domaći programi. Hit meseca je bila avantura SUPI (skraćenica od *Sezam U Prekid*) **Stanislava Šokorca** (*ssokorac* na Sezamu). Oduševljeni igrači su poteklí autora da još istog meseca napiše i nastavak – SUPI 2 – koji je još bolje primljen, naročito u onome delu u kome treba poljubiti sekretaricu Sezama. Sada se traži treći deo. Dobili smo PD simulator pokera za PC računare, od pravih automata se razlikuje samo po tome što ne traži pare, autor je **Predrag Knežević**, a program je poslao *andrej*! **Darko Stanišić** (*darons*) je poslao program – slagalicu po imenu ARGH – ime posleje vrlo razumljivo čim pokušate da rešite neki od problema. Ljubitelji igre *Trade Wars* (nešto kao višekorisnička *Elita*, trenutno verovatno najpopularnija BBS igra na svetu) su razvili živu diskusiju, mada u „malo" neprikladnoj temi – *klub*.

KAKO POSTATI ČLAN

Postupak učlanjenja počinje zvanjem Sezama – podesite parametre komunikacije na 2400 8N1 (ako imate MNP modem, uključite MNP) i pozovite (011)648-880. Posle pozdravne poruke, Sezam će ispisati pitanje

Username:

Na vama je samo da otkucate NEW (novi korisnik), a Sezam će vas dalje voditi kroz proces prijavljivanja.

Izbor imena

Jedan od prvih podataka koje treba da date je puno ime i prezime, a potom i pseudonim pod kojim ćete koristiti usluge Sezama. Jednom izabran pseudonim ostaje vaše trajno ime na Sezamu (ne može se menjati), što znači da ga treba pažljivo izabrati – predlažemo da se još pre poziva opredelite za pseudonim koji bi trebalo da podseća na vaše ime i prezime. Ukoliko nemate ideja, Sezam će vam predložiti pseudonim sastavljen od vašeg prezimena i prvog slova imena. Pseudonim može da ima između 4 i 16 znakova (preporučuje se 5-8 slova, pošto više znakova znači više kucanja, kako za vas tako i za one koji sa vama komuniciraju) i u njemu se mogu naći isključivo slova i znak tačka.

Izbor lozinke

Sledeći korak je izbor lozinke (*password*) kojom štitite vaš račun – za razliku od pseudonima koji je javna informacija, lozinku znate samo vi i Sezam. Zato je izaberite pažljivo, naročito izbegavajte da lozinka bude jednaka pseudonimu ili da se sastoji od vašeg imena, nadimka ili imena nekog člana porodice – takve lozinke se lako pogode, što donosi različite probleme, pre svega onome čija je lozinka „ukradena". Izabranu lozinku kasnije možete da menjate.

Izbor radnih parametara

Ostaje još da izaberete radne parametre (arhiver, protokol i slično – na brzinu ako još ne znate šta je šta, posle će Sezam uvek predložiti uobičajene vrednosti).

Unos ličnih podataka

Prilikom unosa adrese i telefonskog broja budite veoma pažljivi, jer netačno unesena adresa onemogućava kontakt Uprave Sezama sa vama, a samim tim i vaše učlanjenje. Unošenjem adrese i ostalih podataka završena je prva faza vašeg prijavljivanja na Sezam.

Redovno...

Sledećeg radnog dana biće vam poslato pismo sa obaveštenjima o Sezamu i popunjenom uplatnicom. Ukoliko želite da proverite da li su podaci koje ste uneli ispravni, pozovite sledećeg dana, unesite izabrani pseudonim i lozinku. Sezam će potvrditi da vam je pismo poslato ili zahtevati da dopunite podatke. Pošto dobijete pismo, treba da izaberete period na koji se pretplaćujete i na osnovu toga uplatite odgovarajući iznos:

za 1 mesec	1600 dinara
za 3 meseca	3300 dinara
za 6 meseci	5700 dinara
za 12 meseci	9500 dinara

Nakon izvršene uplate fotokopiju potvrde pošaljite pismom na adresu „Računari", Bulevar vojvode Mišića 17, Beograd ili faxom na telefon (011)648-140 ili (011)647-936. Uprava će vam na dan prijema potvrde odobriti pristup sistemu, a sledećeg dana, će na vašu adresu biti poslato Uputstvo za upotrebu Sezama, atraktivno štampana knjižica (160 strana) pomoću koje ćete upoznati sve mogućnosti sistema. Korisnici koji se pretplate na mesec dana ne dobijaju uputstvo dok prvi put ne produže pretplatu.

...i ekspresno učlanjenje

Oni kojima se žuri mogu postati korisnici Sezama za svega nekoliko sati. Javite se na opisan način a onda, ne čekajući pismo, izvršite uplatu u pošti navodeći podatke: **Pošiljalac**: Vaše ime, prezime i pseudonim koji ste izabrali; **Primalac**: BIGZ – Računari; **Svrha uplate**: Pretplata na Sezam; **Žiro račun**: 60802-603-23264, **poziv na broj**: 05 108-5. Potvrdu o uplati pošaljite faxom do 13 časova (radnim danom) i već u popodnevnim časovima ste punopravni korisnik Sezama!

Dvostruko prijavljivanje

Ukoliko tokom predstavljanja imate problema (npr. veza se prekine), pozovite ponovo i prijavite se navodeći kompletne podatke. Treba ipak izbegavati višestruko prijavljivanje jer pismo i uplatnicu šaljemo samo jednom. Pažnja: ako ste se prijavili dva i više puta, koristite ono korisničko ime koje je napisano na uplatnici.

Koliko važi uplatnica

Cene navedene u pismu važe 7 dana – ukoliko ne izvršite uplatu u tom roku, moguće su korekcije, ali to ne znači da se morate ponovo prijavljivati – dovoljno je da pozovete Sezam, predstavite se izabranim imenom i lozinkom i sistem će ispisati ažurne cene pretplate.

Razgledanje sistema

Ukoliko pre prijavljivanja želite da razgledate sistem, na pitanje **Username** odgovorite sa DEMO. Naći ćete se u demonstracionom režimu sada u kome možete da pregledate kompletnu strukturu sistema koja je identična „pravoj". Broj DEMO poziva nije ograničen, a vreme za jednu seansu je 5 minuta.

Dobili smo i dva uputstva za tu igru. U temi *elektron* je razmatran problem dogradnje upravljačkog sistema na strug koji se kontroliše PC-jem. U okviru iste diskusije dobili smo i uputstvo kako se za samo 49 DEM u samogradnji može napraviti ploter. Vlasnici *Sharp*-ovih džepnih računara saznali su da se i njihovi kalkulatori mogu lako povezati sa PC računarima (mnogo je zgodnije pisati programe na velikoj tastaturi i čuvati ih na disketama). U temi *hp48sx* opisane su sve prednosti HP48 nad HP28. Tema *literatura* je pomogla da se nađu knjige posvećene programima *Clipper* i *CorelDraw*. U temi *trendovi* najavljeno je slanje simulatora paralelnog računara (do 64 procesora) za PC. Stigao je još jedan program (izvorni kod u fortranu) za konverziju datuma u julijanski dan i obrnuto. Na samom kraju meseca najavljeno je slanje PD prototipa kompajlera za (više ne baš tako) novi programski jezik Niclaus Wirth – Oberon, „nastavak" paskala i Module 2.

U **FORUM** je tokom avgusta stiglo neuobičajeno mnogo poruka, blizu 1500. Razlog nije samo dramatična situacija u zemlji i okolini, nego pre svega jedna „beskrajna" diskusija o praksi vraćanja knjiga autorima čija vas je politička delatnost razočarala – da li je to demokratska praksa ili predznak fašizma? Druga „velika tema" se odnosila na eventualnu stranu vojnu intervenciju i njene modalitete – kako bi počela, šta bi bilo na udaru i kakve su naše šanse. Diskutovalo se, osim toga, i o poverenju Saveznoj vladi, stezanju sankcija Ujedinjenih nacija, ljudskim pravima u tako zahvaćenim predelima i drugim temama. Krajem avgusta konferencija FORUM je dostigla svoj limit od 5000 poruka pa je postala *read only* – poruke prima nova konferencija, sedma po redu FORUM u trogodišnjoj istoriji Sezama. Sve prethodne konferencije su, prirodno, i dalje na sistemu, tako da se uvek može proveriti čije su se prognoze u kom domenu ostvarile i da li su stvari tekle onako kako su Sezamovci predviđali ili mnogo gore.

Bio je to najtopliji avgust mesec za proteklih 100 godina. Začudo, ovo ne samo što nije smetalo konferenciji **CIVILIZACIJA**, već je, po svemu sudeći, učinilo da se obore svi rekordi. Visoke temperature utiču da pauci povećaju svoju aktivnost, a fina mreža sadržajnih diskusija u gotovo svim temama najviše i podseća na minuciozan rad pauka. Bilo je izazovno pratiti iznenađujuće kvalifikovane rasprave o pitanjima jezika. Bili su veoma korisni, ne baš svima dostupni, i podaci o nama dalekom Japanu. Mnogobrojni učesnici pokazali su veoma široko neslaganje u stavovima, ali je ovakva raznolikost uvek bila veoma konstruktivna s obzirom da je održano ovde uobičajeno poštovanje naše individualne različitosti. Sasvim nova je bila diskusija o trenutnom civilizacijskom trenutku gde su nehotice dodirnuta i temeljna, mogli bismo gotovo reći „filozofske" stanovište o vrednostima i ciljevima života. Ne čudi da su razlike u stavovima ovde bile i najveće. Saznali smo ko su bili dobitnici Nobelovih nagrada sve od daleke 1901. godine. Pitali smo se da li je neophodno striktno uvažavanje gramatičkih pravila u kompjuterskim programima. Još uvek nije pronađen pravi prevod za termin *clipboard*, no vidnu pažnju je privukao šaljivi predlog „stavka". Po ko zna koji put ovlaš je pomenuta Vukova reforma jezika i pisma, i ponovo su stanovišta bila sasvim suprostavljena. Da li će se ove rasprave rasplamsati u septembru ili će sačekati da se i vi uključite, teško je reći. U svakom slučaju, protekli mesec biće teško nadmašiti, ali je činjenica da je ova konferencija, za manje od godinu dana, gotovo dosegla limit od 5.000 poruka što je i za znatno posećenije konferencije na Sezamu odličan učinak. Otvaranje novog tome biće prilika da se eventualno osveži i spisak tema. Imate li neki predlog?

U konferenciji **VICEVI** stanje redovno – iako je zbog letnjeg odmora glasanje za vic meseca odloženo za septembar, konkurencija će biti prilično jaka. Najviše komentara je izazvao jedan prilično star tekst **Matije Bećkovića** objavljen (pod pseudonimom) u (tadašnjem) „ZUM reporteru". Naslov je bio „I žene su ljudi", a među tezama su se pojavljivali i biseri poput: „ako sam novine pročitao, odmah sutradan može da ih čita i ona, pod uslovom da ne komentariše to što pročita", „ako nisam kod kuće može da uključi radio ili TV čim počne jeftine struje", „ako se dete prehladi, sama bira koga da bijem – nju ili njega", „može da pegla i pere prozore kad ona hoće, pod uslovom da uvek sve bude čisto i ispeglano", „našem petom detetu i ona je smela da predloži ime koje zamalo nije usvojeno", „od Nove godine oslobodio sam je obaveza da mi vraća žute metalne pare", „nikada je ne isterujem na ulicu. Samo na terasu" i tome slično. Bilo je, kao i obično, i dosta političkih viceva, posebno na račun raznih naših predsednika, a nije pošteđen ni gospodin ministar prosvete. Za kraj jedan „crnjak" posle višemesečnog boravka u Bosni Kenijac, član UNPROFOR-a, se vraća kući. Prvi utisak: „Čudan neki narod. Kolju, a ne jedu!"

SezamFile

Jovan Bulajić **09**

Priliv novih datoteka na Sezam je i u septembru bio solidan – i pored znatno suženih kanala nabavke, pristiglo je tri megabajta u tridesetak novih datoteka. Zapažen je sve veći priliv programa domaćih autora, a otvoren je i poseban direktorijum za *Clipper* programe.

U septembru je u Sezamove direktorijume pristiglo tridesetak novih datoteka ukupne dužine oko 3 megabajta. Istovremeno, korisnici su više od 7000 puta upotrebili komandu **download** da bi iz direktorijuma preneli oko 540 megabajta podataka.

Ovo sigurno nije loš rezultat, s obzirom da su glavni kanali nabavke programa iz inostranstva i dalje prekinuti. Zahtev za skidanje sankcija koje se odnose na naučno-tehničku saradnju, gde spada i naše uključenje u svetske računarske mreže, nalazi se u trenutno aktuelnom paketu premijera Panića. Ali dok se to ne razreši, izvori novih programa na Sezamu su uglavnom stare zalihe, a ređe Jupak ili „lična kurirska pošta" (kad se neko vrati iz inostranstva i usput donese novi softver). Značajan priliv novih datoteka potiče i od – domaćih programa.

Može se reći da je beznadežnost uslovljena opštom situacijom uticala na povećanje radnog elana domaćih programera. Tako je od programa koji su se našli u nekom od PC direktorijuma od početka juna čak dvadesetak delo domaćih autora. Ovo je sigurno podatak za pohvalu, a nadamo se da će se tako nastaviti i ubuduće.

I na listi najtraženijih datoteka su domaći programi u većini. O nekima smo već pisali ranije, a novitete ćemo opisati u ovom broju. *Scan* i *Clean* su na vrhu top liste, iako ovog meseca nije bilo njihove nove verzije (poslednja je s kraja avgusta), a ni novopridošlih virusa. *BTW Scan* je apsolutni rekorder po broju *download*-a, jer je sa Sezama skinut više od 3000 puta. Ovaj broj se odnosi na sve verzije u protekloj godinu i po dana, od kada se vodi detaljnija statistika.

Tabela 1 TOP 10 datoteka – septembar 1992			
direktorijum	**datoteka**		**opis**
1 \IBMPC\VIRUS	scan95b	zip	McAfee virus detektor 8.7B95
2 \IBMPC\VIRUS	clean95c	zip	Uklanjanje virusa koje SCAN nađe 8.7C95
3 \IBMPC\MISC	tel10	arj	Telefonski imenik (bulaje)
4 \IBMPC\TEST	idefe	zip	Automatsko određivanje parametara IDE diska (Z. Dobrić)
5 \IBMPC\FUN	school	arj	School Quest I (bole)
6 \IBMPC\FUN	supl202	arj	Sezam U Praksi II deo – avantura o Sezamu (ssokorac)
7 \IBMPC\FUN	supl10	arj	Sezam U Praksi I deo – avantura o Sezamu (ssokorac)
8 \COM	sor230	zip	SOR: Sezam Off-Line Reader v2.30
9 \IBMPC\SYSTEM	umb_drvr	arj	Memory manager (286/386) podržava razne chip-setove
10 \IBMPC\ARCHIVER	shez80	arj	Shell za rad sa zip, arj, lha i ostalim arhiverima

Tabela 2 Datoteke u septembru i ukupno				
	nove		ukupno	
direktorijum	broj	dužina	broj	dužina
ARCHIVER	0	0	20	1 298 136
C	1	65.953	12	1.534.907
CLIPPER	4	323.590	11	861 026
COM	3	426 409	35	3 888 525
DEMO	5	579 167	15	1 577 676
DRIVERS	5	711.386	25	2.916 532
EDITOR	0	0	17	1.398 045
FUN	2	180 876	26	2 519 337
GRAPHICS	1	140 402	14	1 488 918
INFO	3	111 532	23	2.815 730
MISC	1	49 480	33	3.459 208
PROGRAM	0	0	16	1 547 458
SYSTEM	1	40 130	17	1 389 484
TEST	0	0	13	1 120 451
UTILITY	2	34 641	50	2 212.002
VIRUS	0	0	15	1 592 793
WINDOWS	4	317.363	31	2 791 631
YU	0	0	13	297 581
Ukup IBMPC	32	2.980 929	386	34 789 484
OS2	0	0	12	788 905
RSOFT	1	25.703	50	612 468
UNIX	0	0	27	2.982 877
Ukupno	33	3.006 632	475	38.151 698

Domaći programi

Mnogi od vas sigurno koriste računar između ostalog i za smeštanje raznih svakodnevno potrebnih podataka, npr. telefonskih brojeva. Ovo je verovatno jedna od najčešćih zabava za novopečene korisnike računara, pa onda obično srećno pokazuju komšiji šta sve može njihova nova mašina. Na Sezamu se u septembru našao i jedan naš program za vođenje telefonskog imenika. Jednostavan je za korišćenje, omogućava standardne opcije unosa i pretraživanja podataka, podržava rad sa više baza podataka i koristi standardan DBF format datoteka. Nama se najkorisnijom čini mogućnost pretraživanja iz DOS komandne linije. Program je naših ruku delo (...) i može se naći u **\ibmpc\misc** direktorijumu kao **tel??.arj** (?? je oznaka poslednje verzije, trenutno je u pripremi nova).

Da Sezam ima značajnog uticaja na širenje računarske kulture za mnoge nije sporno – one koji su sumnjičavi će razuveriti i najnoviji prilog domaćeg autora. Jedan od najmlađih korisnika Sezama, **Boško Radivojević** *(bole)* je, očigledno inspirisan prilozima daleko starijih korisnika, napisao kompletnu igru – avanturu čija se radnja odvija u krugu njegove škole *(School Quest II)*.

Poker v2.0 je (kao što mu i ime kaže) program koji predstavlja simulaciju jednog od brojnih poker aparata. Podržane su uglavnom standardne mogućnosti jedne ovakve mašine. Program radi sa standardnim grafičkim karticama (Herc/CGA/EGA/VGA). Možete ga naći u **\ibmpc\fun** direktorijumu kao **poker200.arj**, a autor je **Predrag Knežević**.

Uz novu verziju *FDS-a 2.04*, **Miljan Jovanović** *(mjova) je poslao još jedan zgodan uslužni (utility)* program za pregled stanja na vašem disku. SCD *(Scan Dir Space)* ispisuje zbirnu statistiku datoteka po poddirektorijumima, uz prikaz stvarnog zauzetog prostora *(allocated)*. Kao i uz FDS, i uz ovaj program je dat izvorni kod, pisan u C-u. Možete ga naći u **\ibmpc\utility** direktorijumu kao **scd171.arj**.

U „Računarima 84" ste čitali prikaze nekoliko YU programa, a na Sezamu možete naći demo verzije za dva od njih (u direktorijumu **\ibmpc\demo**) – *Raspored 2.0* (autor **Predrag Marković**) i *Instant Biro 1.0* (autor **Igor Čuić**, mikroLINK).

Clipper

Clipper programeri su veoma brojni na Sezamu, sudeći po tome kakvo interesovanje vlada za datoteke vezane za ovaj kompajler. Njih moramo priznati još uvek nema u zadovoljavajućem broju, ali pošto nam je traženje dobrih PD stvari za *Clipper* (ij. uopšte za programere) jedna od prioritetnih operacija, biće bolje. Zato smo prošlog meseca otvorili i poseban direktorijum **Clipper** u IBMPC grani, u kome se odmah našlo i nekoliko zapaženih noviteta.

O nekim *Clipper* programima koji se mogu naći na Sezamu je već pisano u „Računarima" – **NanForum Toolkit** biblioteka (NFLIB) raznih funkcija za koje je dat i izvorni kod (imate članak u R80) i **O-Clip** objektno orijentisane proširenja *Clipper*-a (R82) su zaista stvari koje vredi pogledati.

Novi programi za *Clipper* u septembru bili su nova verzija *Clip-Graph* grafičke biblioteke, **Expand Library** u kojoj se nalaze razne funkcije, izvorni kod za implementaciju skrolujućih GET-ova i biblioteka koja omogućava selektivno filtriranje slogova u .NTX indeksnim datotekama (što je za tiksne uslove neuporedivo brže od **Set filter** naredbe).

Clip-Graph 2.1 (ranije je na Sezamu bila verzija 1.2) je biblioteka za rad sa grafikom i mišem za *Clipper* verzije 5.01 i S87. Omogućava osnovne komande u grafičkom modu – crtanje linija, jednostavnih geometrijskih oblika, ispisivanje teksta, podržava sve standardne grafičke kartice, a omogućava i prikaz PCX datoteka.

Expand Lib 2.0 je biblioteka koja sadrži funkcije raznih namena za *Clipper* 5.01 – za rad sa diskom, datotekama, mišem, štampačem, ekranom, Lotus 1-2-3 datotekama, DOS okolinom, datumima, prepoznavanje hardverske konfiguracije.

Uz datoteke koje se nalaze u **\ibmpc\clipper**, tu su i brojni odlični primeri uz tekstove u „Računarima" koje možete uzeti iz RSOFT direktorijuma.

Ostali noviteti

UMB DRVR je program za upravljanje memorijom koga korisnici Sezama dosta hvale, a nalazi se i među deset najtraženijih programa prošlog meseca. U dokumentaciji ga nazivaju „UMB provider for DOS 5.0 on 286/386/486 systems", a omogućava više stvari – proširuje DOS memoriju iznad 640KB (ako je moguće), pravi UMB (*Upper Memory Blocks*) za učitavanje programa i drajvera u gornju memoriju, pravi *shadow* ROM-ove. Podržava veliki broj raznih čip-setova (npr. kod nas raširene C & T i OPTi, a podrška za Headland se najavljuje u sledećoj verziji), a u paketu se dobija i EMS 4 emulator. Za UMB_DRVR kažu da nije baš kao QEMM, ali da je bolji od QRAM-a, pa nije ni čudo što su korisnici 286 računara bili nezainteresovani za ovaj drajver.



Nenad Batoćanin



IMAM RAČUNAR- ODAKLE DA POČNEM?

SPECIJALAN DODATAK ČASOPISA *računari*
novembar 1992

Posvećeno svim mojim prijateljima koji mi neprestano postavljaju ista pitanja.

Konačno ste kupili računar! Posle beskonačnog proučavanja oglasa, telefoniranja na uvek zauzete brojeve, razgovora i saveta, čitanja umetka iz „Računara" o izboru konfiguracije, magična kutija se najzad našla na vašem stolu. Ako znate od ranije da programirate ili ste koristili neki program, vaši glavni problemi su uglavnom završeni. Međutim, ako niste nikad radili na računaru, znajte da vaši problemi tek počinju!

Naredni tekst je upravo vama posvećen. U njemu će biti govora o osnovnim stvarima: kako radi računar, kako rade programi i kako sve to na pravi način iskoristiti. Naravno, u ovako kratkom tekstu ne može biti objašnjeno sve što će vam ikad zatrebati; cilj je da shvatite osnovne pojmove bez kojih ne možete koristiti ni jedan program a ni priručnik. Kada savladate te osnovne stvari, moći ćete mnogo lakše da pratite tekstove u „Računarima", moći ćete da koristite mnoge programe – ukratko, bićete sposobni da dalje napredujete.

Tekst nije pisan na suviše „niskom" nivou: neće biti previše ilustrativnih poređenja i sl. Ovo je jedan sažet kurs o osnovnim pojmovima koji ima za cilj da vam ukratko objasni osnovne stvari, ili da vam raščisti neke dileme oko njih. Neki delovi ovog teksta nisu pisani za apsolutne početnike, pa možda nećete moći da ih razumete prilikom prvog čitanja. Međutim, to su veoma praktične i veoma upotrebljive stvari koje će vam zatrebati kada samo malo budete koristili računar. Iz tog razloga su i uključeni u ovaj tekst. Dobar deo teksta računa na vašu radoznalost i sposobnost zaključivanja, tako da nije sve uvek do detalja objašnjeno, ali u takvim delovima ćete se sasvim sigurno uz minimalan napor i sami snaći.

RAČUNAR I OPERATIVNI SISTEM

Bez obzira kako vam komplikovano i moćno izgledao, računar je ipak samo jedna obična mašina – velikih mogućnosti i jako korisna, ali ipak samo mašina. Računari se prave uz pomoć najsavremenije tehnologije, kao i ostale elektronske naprave, video, televizor. To znači da imaju veoma komplikovanu unutrašnjost i vrlo je komplikovano objasniti **detaljno** kako radi računar. Međutim, to vam i nije potrebno: za korišćenje računara potrebno je razumeti samo osnovne principe rada, a koga interesuje više može naći detalje u odgovarajućim knjigama. Čak, za uspešno korišćenje skoro da nije potrebno znati **kako** računar zaista radi, ali je svakako preporučljivo to bar donekle razumeti.

KAKO RADI RAČUNAR?

Da bi lakše razumeli kako radi računar, zanemarićemo mnoge detalje u njegovoj strukturi: na slici 1 prikazana je **veoma** grubo od čega se sastoji računar. Ono što možete od prikazanih delova videti i bez otvaranja kutije računara, to su monitor i tastatura. Ostali delovi se nalaze u kutiji računara.



1. Tastatura, monitor i računar, gruba skica principa rada

Kako sve to radi? Računar možete posmatrati kao jednu mašinu koja „zna" da izvodi određene operacije. Koje će to operacije biti određuje korisnik korišćenjem tastature. Kucanjem specijalnih instrukcija na tastaturi korisnik naređuje računaru **šta** (koju operaciju) treba da uradi. Za taj „rad" (kaže se „obrada podataka", jer računar obrađuje podatke) zadužen je deo računara koji se zove **procesor**. Kada procesor primi naređenje da nešto uradi, on to pokuša da uradi (jer se može desiti da ne može to da izvrši), a o rezultatu te operacije obaveštava korisnika na ekranu monitora. Tada se vidi neka poruka poput „Nisam uspeo da izvršim naredbu" ili nešto slično. Na osnovu te poruke korisnik dalje odlučuja šta će dalje preduzeti, naređuje novu operaciju itd. Na taj način korisnik komunicira sa računarom.

Znači, procesor je, u neku ruku, glavni deo računara: on obrađuje podatke i na osnovu toga daje rezultate. U vašem računaru procesor je jedan mali elektronski sklop – ako otvorite kutiju prepoznaćete ga po oznaci 8086, 80286, 80386 ili 80486. To su različiti modeli procesora i jedina razlika koju ćete u početku videti između njih je brzina: na primer, 80486 je mnogo brži od 8086 procesora. Postoji još razlika, ali vama je za sada najvažnija brzina.

Procesor je sposoban da izvršava veoma jednostavne operacije sa brojevima: da sabere ili oduzme dva cela broja, da uporedi dva broja, itd. Ovakve operacije su čoveku sasvim nepotrebne, ali kada se grupiše više ovakvih instrukcija, dobija se nešto sasvim korisno. Procesor istovremeno može da obrađuje sasvim malo podataka: jedva nekoliko brojeva. Pošto je za praktičnu korist potrebno obraditi daleko više podataka, računar sadrži **operativnu memoriju**: elektronski sklop koji je zadužen za čuvanje podataka kada ih procesor ne obrađuje. Znači, procesor uzima podatke iz memorije, obrađuje ih i vraća natrag obrađene podatke. Memoriju možete zamisliti kao niz ćelija, pri čemu svaka pamti jedan broj. Svaka ta ćelija se naziva **bajt**. Količina podataka koja se može upisati u memoriju naziva se **kapacitet memorije**. Memorija u vašem računaru verovatno ima kapacitet od milion i više bajtova. Pošto je ovo prilično nepregledno, uvode se veće jedinice za kapacitet: **kilobajt** (1024 bajta) i **megabajt** (1024 kilobajta). To znači da operativna memorija u vašem računaru ima kapacitet od jednog megabajta (piše se 1MB) ili 1024 kilobajta (1024K).

Instrukcije po kojima procesor obrađuje brojeve takođe se čuvaju u operativnoj memoriji: svaka instrukcija je u stvari jedan broj i ne postoji neka razlika između podataka i tih instrukcija u memoriji. Procesor redom čita takve instrukcije i izvršava ih jednu po jednu. Niz takvih instrukcija čini **program**. Znači, i program se kao i podaci čuva u operativnoj memoriji.

U memoriji se čuvaju samo brojevi – ali, ti brojevi se mogu posmatrati i kao neki drugi podaci: uzećemo da je broj 65 slovo „A", broj 66 slovo „B", itd. Na taj način se u memoriji mogu predstavljati i razni simboli, pa i ostale vrste podataka.

Ma kako vam izgledao veliki kapacitet memorije od milion bajtova, videćete da je to prilično malo, jedva za jedan ili dva veća programa. A pošto vi želite da koristite više različitih programa, morate ih imati pri ruci mnogo više. Ova memorija ima još jednu manu: podaci u njoj se čuvaju samo dok je računar uključen. Čim se računar isključi, podaci se nepovratno gube.

Zbog svega toga vaš računar ima i **disk**, to je mehanička sprava koja slično operativnoj memoriji čuva podatke, ali ima bitne prednosti: na disku se može čuvati mnogo više podataka, podaci ostaju i kad se isključi računar, a i mnogo je jeftiniji megabajt ovakve nego elektronske memorije. Disk je daleko sporiji od elektronske memorije. Podaci su malo drugačije organizovani na disku nego što su u memoriji, ali to nama nije bitno. Važno je da znamo da se podaci sa diska učitavaju u memoriju, tu se obrađuju, a zatim se ponovo mogu upisati na disk. Time je omogućeno da na jednom računaru koristimo mnogo programa: ali se uvek izvršava samo jedan: taj se učita sa diska u memoriju i onda ga procesor izvršava. Vaš disk obično ima kapacitet od 20, 40 i više megabajta.

Vaš računar sigurno ima i disketnu jedinicu: to je sprava sasvim slična disku, ali mnogo sporija i sa mnogo manjim kapacitetom. Međutim, disketna jedinica služi za unos podataka u vaš računar: sa diskete se podaci snimaju na disk i obratno se koriste. Takođe vi možete snimiti podatke sa vašeg diska na disketu i tako preneti podatke na neki drugi računar. Tako je disketna jedinica (često se kaže samo **disketa**) neka vrsta prozora u svet vašeg računara.

Osim nabrojanih delova, računar može biti povezan sa velikim brojem raznih uređaja. U nastavku su nabrojani neki od najvažnijih, sa kojima ćete se verovatno sresti.

Štampač – Štampač ima sličnu namenu kao i ekran: na njemu se vide rezultati obrade podataka, s tim što se ovde podaci dobijaju na papiru, što je velika prednost. Štampač se koristi uvek kada je potrebno rezultate rada nekih programa zabeležiti trajno i predati nekom ko te rezultate neće računarski obrađivati (slika 2).

2. Štampač

Ploter – Ploter je uređaj koji služi za crtanje slika. Slika se prvo nacrta specijalnim programom na računaru, a onda se uz pomoć odgovarajućeg programa crta slika na ploteru.

Miš – To je uređaj koji služi za unos podataka. Pomeranjem miša po nekoj podlozi pomera se odgovarajući pokazivač na ekranu. Na taj način korisnik može mnogo preciznije i brže da pokaže neku poziciju na ekranu nego komandama sa tastature. Obično se koristi u programima za crtanje (slika 3).

3. Miš

Traka – Ovo je uređaj koji zamenjuje diskete. Sličan je standardnom kasetofonu i služi za čuvanje većih količina podataka.

Modem – Pomoću ovog uređaja možete svoj računar povezati sa drugim računarima širom sveta korišćenjem telefonskih linija. Postoji veliki broj računara širom sveta koji sadrže ogromne količine veoma korisnih informacija koje možete

dobiti podaci o kretanju na tržištu, najnovije informacije, reklame, oglasi, itd. Jedan od takvih računara (u stvari, više računara) je i „Sezam" časopisa „Računari".

Skener – Ovim uređajem možete sliku uneti u računar. Pod „slikom" se podrazumeva dosta različitih stvari: crteži, fotografije, pa i tekst. Uneta slika se može obrađivati specijalnim programima ili uključivati u tekstove što je zgodno za razne ilustracije. Treba imati u vidu da se ovim kvalitet slike gubi, pa za dobre rezultate morate kupiti veoma kvalitetan skener.

OPERATIVNI SISTEM

Kao što smo videli, procesor je sposoban da i zvršava samo veoma jednostavne, i čoveku besmislene operacije. Instrukcije koje treba zadati procesoru su neki brojevi, a oni korisniku računara ništa ne znače. Međutim, kada se procesoru zada niz takvih instrukcija – brojeva (u stvari, taj niz je **program**), procesor može uraditi neku sasvim korisnu stvar: bilo koji program koji ste koristili je samo jedan niz brojeva. Taj niz brojeva se čuva na disku, baš kao i bilo koji podatak, zatim se učitava u operativnu memoriju odatle procesor uzima jedan po jedan broj i izvršava time određene operacije.

Naravno, bilo bi suviše očekivati i od profesionalnog programera da dobro razume taj jezik procesora (zove se **mašinski jezik**), a kamoli od korisnika računara. Zato se uvodi jedan program koji stalno radi i koji ima ulogu posrednika između računara (procesora) i korisnika računara. Taj program se naziva **operativni sistem**.

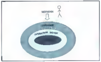
Kada se računar uključi, prvo se u memoriju učitava sa diska taj program – operativni sistem. Kada korisnik otkuca neku naredbu, šalje je operativnom sistemu, on je transformiše i prosleđuje dalje procesoru (ponekad i drugim delovima računara). Kada procesor izvrši naredbu (u stvari, izvršava čitav niz mašinskih instrukcija koje realizuju jednu zadatu komandu), on šalje operativnom sistemu poruku o rezultatima obrada. Operativni sistem to prevodi na jezik koji mi razumemo i na neki način nam to saopštava, najčešće ispisuje neku poruku na ekranu i sl.

Na taj način korisnik može da koristi resurse računara, bez poznavanja jezika procesora ili detalja oko unutrašnjosti računara: on jednostavno komunicira sa operativnim sistemom, a ostalo nije njegova briga. Ovo je opšti princip koji važi i na drugim računarima, na samo na *PC*-u kakav vi imate. Operativni sistem vašeg računara se naziva *MS-DOS*, a često se naziva i samo *DOS*. To je skraćenica od *Microsoft Disk Operating System* (*Microsoft* je firma koja je napravila *MS-DOS*).

PROGRAMI

Operativni sistem ima prilično skromne mogućnosti: on sadrži komande pomoću kojih možete izbrisati ekran, izvesti neku prostu operaciju sa podacima na disku i sl. Pravu moć računaru daju drugi programi. Kao što je već rečeno, strogo gledajući, programi su obični nizovi brojeva koje razumeju jedino procesor i operativni sistem. Međutim, ti nizovi brojeva izvode veoma korisne akcije i oni su upravo ono što računarima daje glavnu moć.

Pre nego što se aktiviraju, programi se čuvaju na disku. Kada naredite aktiviranje nekog od njih, program se sa diska učitava u memoriju, odakle ga procesor instrukciju po instrukciju izvršava. Programi koriste operativni sistem kao posrednika za rad sa računarom, pa konačna situacija izgleda kao na slici 4.



4. Sistem korisnik-program operativni sistem računar

Kada korisnik zadaje razne komande programu, program te komande analizira i preko operativnog sistema šalje odgovarajuće instrukcije procesoru. Procesor obrađuje podatke i rezultate šalje preko operativnog sistema programu. Program to ponovo analizira i na kraju šalje korisniku programa izveštaj o obradi, posle čega ponovo korisnik zadaje svoju komandu, itd.

Danas na tržištu postoji ogroman broj programa koji vam omogućavaju da vršite razne obrade podataka: vodite knjigovodstvo za firmu, pišete razne tekstove, crtate slike ili se jednostavno igrate. Može se slobodno reći da programi postoje za skoro svaku oblast gde nije potrebna intuicija i inteligencija čoveka. Zato sve dosadne poslove slobodno prepustite računaru: on će to sigurno mnogo kvalitetnije i brže uraditi od vas! Računar je veoma brza mašina, pa u nekim slučajevima nedostatak ljudske inteligencije može nadoknaditi sirovom snagom: postoje izuzetni programi koji igraju šah i slične igre, ali to nikako ne znači da su računari postali „pametni" niti će to uskoro postati. U suštini, može se reći da je računar „pametan" koliko i program napisan za njega.

Postavlja se pitanje, šta je od programa vama potrebno? Najveća greška je ako počnete da nabavljate programe bez ikakve selekcije. Na ovaj način ćete veoma brzo popuniti prostor na disku, a postoji i opasnost da sa tim programima preko diskete prenesete i programe koji mogu naškoditi podacima i drugim programima: takozvane **viruse**. Postoje posebni programi koji otkrivaju prisustvo ovih virusa na disketama, i jedan od njih svakako nabavite.

Sve u svemu, prvo morate razmisliti i odlučiti šta ćete raditi sa računarom. Najbolje je da o tome porazgovarate sa nekim iskusnijim. Tek onda nabavite odgovarajuće programe i nastavite sa radom: sumnjive programe nikako nemojte stavljati na vaš računar.

Programi za PC računare su napravljeni za veoma veliki broj praktičnih primena. Skoro je nemoguće nabrojati sve te oblasti, ali postoje neke koje su najvažnije i najpopularnije. Podela nikako nije stroga: mnogi programi se mogu staviti i u nekoliko grupa odjednom. Evo najvažnijih grupa programa.

Pomoćni programi – Programi iz ove grupe služe da vam olakšaju neke svakodnevne operacije sa računarom: kopiranje datoteka i disketa, podešavanje raznih parametara i sl. Najpoznatiji programi iz ove grupe su *PC-Tools*, *Norton Utilities*, itd.

Programi koji otkrivaju viruse – To su programi koji otkrivaju viruse na disketama, pre nego što ih prenesete na disk i tako „zarazite" ostale programe. Ako vam je računar već „zaražen", oni vam pomažu da ih otkrijete i eventualno uklonite. Najpoznatiji program iz ova klase je *SCAN*. Ovaj program se može besplatno nabaviti, između ostalog i na nekom od „javnih" računara. Možete ga nabaviti i na „Sezamu".

Obrada teksta – Kucanje teksta je mnogo jednostavnije na računaru nego na standardnoj mašini. Na računaru možete proizvoljno menjati tekst, ispravljati greške i tek na kraju ga odštampati na štampaču. Osim ovoga možete veoma jednostavno pisati cirkularna pisma, crtati tabele i sl. Najpoznatiji programi ovog tipa su *WordPerfect*, *WordStar*, kod nas popularni *ChiWriter*, itd. Postoje i programi namenjeni profesionalnoj pripremi knjiga za štampu. Iz te klase kod nas je najpopularniji program *Ventura*.

Crtanje – Na računaru možete korišćenjem specijalnih programa crtati veoma složene slike. Kasnije se ove slike mogu uklopiti u tekst i odštampati pomoću štampača ili plotera. Najpoznatiji predstavnici ovih programa su *CorelDraw*, *PaintBrush*, *DrawPerfect*, itd. Postoje i specijalni programi koji služe za izradu animacija i profesionalnih tehničkih crteža.

Radne tabele – Ova klasa programa omogućava da radite sa tabelama raznih podataka. Kada popunite tabelu, možete izvoditi razne operacije: sabrati dve kolone, crtati razne grafikone na osnovu ovih podataka i sl. Najpoznatiji program iz ove klase je *Lotus 1 2-3*.

Baze podataka – Ovi programi vam omogućuju da radite sa velikim količinama podataka. Možete da upisujete nove, da pregledate i menjate stare podatke, da generišete razne izveštaje i sl. Najpoznatiji proizvod iz ove klase je program *dBASE*.

Poslovna obrada podataka – Na našem tržištu postoji veliki broj firmi koje prodaju programe za vođenje knjigovodstva i razne poslovne obrade podataka. Izbor je zaista veliki, ali pri tom treba imati jednu činjenicu u vidu: postoje dve vrste ovakvih programa. Prvo su standardni programi koji se prodaju u većim tiražima i oni su znatno jeftiniji. Drugu vrstu čine programi rađeni po zahtevu određenog korisnika i oni su znatno skuplji. Zato morate da se raspitate za programe koji su vam potrebni: ako ne nađete odgovarajući gotov program, morate naći firmu koja će vam uraditi program po narudžbini.

KAKO IZABRATI PROGRAME?

Od velikog broja programa vama zaista trebaju svega nekoliko, bar u početku. Zato nabavite samo programe neophodne za rad – nemojte nepotrebno gomilati razne programe. Već ste videli da to predstavlja potencijalnu opasnost za ostale podatke na vašem računaru, a i nepotrebno zauzima prostor.

Izbor pravih programa nije nimalo lak zadatak čak ni za one koji sa veoma dugo bave računarima, za jednu namenu postoji veliki broj veoma kvalitetnih programa, svi imaju neke prednosti i mane, pa je izbor ponekad veoma težak. Ipak, neka klasifikacija se može napraviti. Naime, postoje određeni programi koji imaju najviše mogućnosti, oni koji imaju manje mogućnosti, ali su nešto jednostavniji za upotrebu, i programi malih mogućnosti. Tako početni skup delimo na tri dela. No, prvo treba odlučiti za šta ćemo uopšte koristiti računar i koji su nam programi potrebni.

Osnovno što vam je potrebno je operativni sistem – to je već pomenuti *MS-DOS*. Verovatno ste ga dobili već instaliranog na računar i tu nije potrebno ništa dodavati.

Kao prvi dodatni program treba nabaviti antivirusni program *SCAN* koji možete dobiti besplatno: raspitajte se i sigurno ćete ga naći.

Neke operacije korišćenjem *MS-DOS*-a mogu biti veoma zamorne, neke čak i nemoguće. Zato je dobro imati na disku i neki pomoćni program koji omogućuje razne operacije. Za te svrhe se u

glavnom koristi program *PC Tools* i to verzija 4.3 – ona je mnogo manja i praktičnija od novijih izdanja (verzija) ovog paketa. Od pomoćnih programa dobro je nabaviti i paket programa *Norton Utilities*. Ti programi vam omogućuju neke operacije sa podacima na disku i disketama, a mogu biti korisni kod problema sa diskom. I to je sve što vam treba od osnovnih alatki. Sve ostalo je stvar vašeg izbora.

Izbor programa za pisanje i obradu tekstova na računaru je nešto osetljiviji problem. Dakle, koji program za pisanje valja uzeti? Verovatno će vam svako na ovo pitanje drugačije odgovoriti svako po svojim navikama i ukusu uzima odgovarajući program. Za početak nećete mnogo pogrešiti ako uzmete bilo koji, jer će u početku svi zadovoljavati vaše potrebe. Međutim, kako budete napredovali, vaše potrebe će rasti pa ćete kod jednostavnijih programa brzo doći do tačke sa koje se dalje ne možete pomeriti. Zato je preporučljivije da odmah uzmete program najviše klase. možda će vam u početku i biti malo teže, ali će vam se trud ubrzo isplatiti.

U ovom tekstu će biti reči o jednom od najmoćnijih i najpopularnijih programa za obradu teksta *WordPerfect* u. Ipak, treba reći da nećete pogrešiti ako uzmete bilo koji program iz ove klase: *WordStar*, *Word*,...

Sve u svemu, za početak možete instalirati sledeće programe na vaš računar: *SCAN* za otkrivanje virusa (nađite što novije izdanje), *PC-Tools v4.3* i *Norton Utilities v4.5* kao pomoćne alatke, *WordPerfect v5.1* za obradu tekstova. Instalacija nekih od nabrojanih programa na vaš računar može biti dosta komplikovana. Zato u narednom tekstu neće biti govora o tome – to je posebna tema o kojoj možete naći dosta tekstova u starijim „Računarima". Za početak, bilo bi dobro da zamolite nekog iskusnijeg da vam instalira ove programe.

Videli smo da je *MS-DOS* naziv operativnog sistema vašeg računara i da on obavlja ulogu posrednika između vas i mašine. Sada ćemo se upoznati sa osnovnim stvarima o *MS DOS*-u.

PROMPT

Kada uključite računar, na ekranu će se ispisati razne poruke i na kraju će se pojaviti tzv. **prompt**. To je niz simbola koji operativni sistem uvek ispisuje kada očekuje vašu komandu. Na primer, prompt može biti **C>** ili **C:\>** ili nešto slično – on se može promeniti, ali je značenje uvek isto.

Posle prompta se ispisuje trepćuća crtica koja se naziva **kursor**. Ona vam pokazuje poziciju na kojoj će se ispisati sledeći simbol kada ga otkucate.

Kada se odlučite za komandu koju želite da zadate računaru, otkucaćete tu komandu slovo po slovo i na kraju pritisnuti taster **<Enter>**. Ovo je veoma značajan taster koji se često koristi – on uvek služi za oznaku kraja unosa komande. Ako ste pogrešili u kucanju, možete izbrisati poslednji ukucan simbol pritiskom na taster **<BS>** (obično je obeležen kao **<–** i nalazi se iznad tastera **<Enter>** na tastaturi).

Kada otkucate komandu, *MS DOS* pokušava da je izvrši. Ako pogrešite u kucanju i zadate neku nepostojeću komandu, operativni sistem će vam ispisati poruku **Bad command or filename**, čime vas obaveštava da ste zadali pogrešnu komandu.

Neke komande su veoma jednostavne. Na primer, komanda **CLS** briše ekran: ako zadate ovu komandu, ekran će se izbrisati i u gornjem levom uglu će se ponovo pojaviti prompt.

DATOTEKE I KATALOZI

Na vašem disku se nalazi veliki broj podataka, tu su programi, razni podaci, itd. Da bi se olakšalo snalaženje u toj masi raznih podataka, podaci su organizovani u **datoteke**. Svaka datoteka sadrži neke podatke (to može biti i program!) i ima svoje ime po kome je razlikujemo od drugih.

Ime datoteke se sastoji iz dva dela, koji se razdvajaju tačkom. Prvi deo je tzv. naziv datoteke, a drugi ekstenzija. Na primer, **TEST.TXT** je naziv (ime) datoteke. Naziv može imati najviše osam simbola, a ekstenzija najviše tri. Možda se pitate, čemu služi ekstenzija kad postoji naziv? Ekstenzija služi za određivanje tipa podataka koji sadrži datoteka. Na primer, datoteke sa ekstenzijom **EXE** sadrže programe, **TXT** tekstove i sl. Pri kreiranju datoteka autor bira i naziv i ekstenziju, pa može staviti šta god hoće, ali veoma je preporučljivo koristiti neke standardne skraćenice.

Datoteke možete porediti sa knjigama u biblioteci – svaka knjiga sadrži neke podatke i ima svoj naziv. Ekstenzija bi mogla predstavljati žanr knjige. Međutim, biblioteke i knjige raspoređuju po grupama zbog lakšeg rada, tako možete naći odeljak za naučnu fantastiku, matematiku i sl. Sasvim slično se na disku organizuju i datoteke: datoteke se grupišu po **katalozima** (ili direktorijumima, kako se često kaže). Svaki katalog sadrži u sebi više datoteka i ima svoje ime. To ime obično označava kakve datoteke se nalaze u tom katalogu: **TEKST** sadrži tekstove, **DOS** sadrži programe *MS DOS*-a itd.



5. Primer stabla kataloga

I tu nije kraj: da bi se olakšalo snalaženje po katalozima, svi katalozi se organizuju u obliku stabla (jedan primer takvog stabla dat je na slici 5). Da bi radili sa datotekama u katalogu, moramo se prvo upoznati sa jednim važnim pojmom: puno ime kataloga i datoteke. Više različitih kataloga mogu imati iste imena. Zato se na osnovu njihove pozicije u stablu određuje njihovo puno ime. Katalog u „korenu" stabla ima naziv \ i to se ne može menjati – vi birate sve ostale nazive. Nazivi ostalih kataloga se dobijaju kada krenete od osnovnog i „nižete" nazive kataloga na koje naiđete. Međusobno ih razdvajate simbolom \. Katalozi na slici koji su prvi naslednici nazivaju se **\TEXT**, **\DBASE** i **\UTIL**, pun naziv kataloga **NORTON** je **\UTIL\NORTON**.

U svakom od ovih kataloga nalaze se neke datoteke. Pun naziv datoteke dobija se kada se na naziv kataloga u kome se nalazi doda naziv datoteke. Na primer, pun naziv datoteke **TEST.TXT** u katalogu **WP** je **\TEXT\WP\TEST.TXT**. U svakodnevnom radu obično se koristi skraćeni oblik, mada je ponekad važno znati i pune nazive.

Sada možete isprobati i jednu komandu *MS*

DOS-a, otkucajte **DIR**, pritisnite taster **<Enter>** i dobićete na ekranu izveštaj sličan ovom:

```
C> DIR
 Volume in drive C is SYSTEM_DISK
 Volume Serial Number is 162F-1805
 Directory of C:\DOS

              <DIR>       07-14-92   7:00p
              <DIR>       07-14-92   7:00p
ATTRIB   EXE     15796 04-09-91   5:00a
PRINT    EXE     15658 04-09-91   5:00a
RECOVER  EXE      9146 04-09-91   5:00a
DISKCOMP COM     10652 04-09-91   5:00a
PRINTER  SYS     18604 04-09-91   5:00a
REPLACE  EXE     20226 04-09-91   5:00a
        9 file(s)      88151 bytes
                     6610304 bytes free
C>
```

Ovom komandom (**DIR**) zatražili ste od operativnog sistema da vam pokaže nazive svih datoteka koje se nalaze u tekućem katalogu. Računar na to odgovara osnovnim podacima o disku, nazivu tekućeg kataloga i ispisuje naziva datoteka sa osnovnim podacima o zauzetoj memoriji, vremenu poslednje izmene, itd. Ako je datoteka označena sa **<DIR>**, radi se o katalogu – nasledniku tekućeg. Katalozi označeni sa „." i „.." su posebni (sistemski) i označavaju tekući (.) i prethodni (..) katalog u stablu.

RAD SA KATALOZIMA

U jednom trenutku možete raditi samo sa jednim katalogom, što i izgleda prilično logično. Kada se računar uključi, nalazite se u osnovnom katalogu (\). Posebnim komandama možete birati i ostale kataloge i onda raditi i sa datotekama u njima. Za izbor tekućeg kataloga služi komanda **CD** (skraćenica od *Change Directory* – promeni katalog). Ukoliko želite da promenite tekući katalog, jednostavno otkucajte **CD**, otkucajte razmak (blanko) i zatim otkucajte koji katalog želite da proglasite za tekući. Ovakve i slične komande dosta često ćete upotrebljavati. Posle naziva komande (**CD**) dolazi jedan ili više tzv. **argumenata**. Argumenti za od komande i međusobno razdvajaju simbolom blanko (razmak).

Na primer, kucanjem naredbe **CD \CRTANJE\DRAW** tekući katalog postaje **\CRTANJE\DRAW**. Naravno, uslov za uspešno izvršenje ove komande je da katalog **\CRTANJE\DRAW** postoji. U zavisnosti kako je podešen vaš sistem, možda će prompt pokazati koji je sad katalog tekući. Ukoliko otkucate samo **CD**, računar će ispisati naziv tekućeg kataloga. Sada možete komandom **DIR** proveriti i sadržaj tekućeg kataloga, tj. datoteka koje su u njemu.

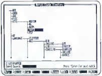
Kada vam stignu novi programi ili počnete rad na nekom novom zadatku, datoteke koje pri tom nastaju treba staviti na neko odvojeno mesto. Dakle, potreban vam je novi katalog. Za kreiranje novog kataloga zadužena je komanda **MD** (od *Make Directory* – kreiraj katalog). Ona se slično koristi kao i **CD**: treba samo navesti kao argument naziv novog kataloga i novi katalog je kreiran. Na primer:

```
MD \TEKSTOVI\PROBA
```

kreira novi potkatalog kataloga **TEKSTOVI**. Ne može se direktno kreirati nekoliko kataloga u stablu: ova komanda će raditi samo ako je prethodno kreiran katalog **\TEKSTOVI**.

Kataloge koji su prazni možete izbrisati komandom **RD** (od *Remove Directory* – ukloni katalog). Na primer, upravo kreirani katalog možete izbrisati komandom **RD \TEKSTOVI\PROBA**. Ukoliko katalog ima datoteke u sebi, prvo morate izbrisati sve datoteke (o tome kasnije), a tek onda katalog.

Ovakav rad može biti veoma zamoran: stablo kataloga je obično veoma razgranato i kucanje dugih naziva uopšte nije lak posao. Zato postoje programi koji olakšavaju ovaj rad. Jedan od takvih je i **NCD** iz paketa *Norton Utilities*. On veoma pregledno zamenjuje prethodno opisane komande. Startuje se tako što otkucate komandu **NCD**. U zavisnosti od verzije koju imate, na ekranu će se pojaviti slika poput one na slici 6 ili slična. Slika koju **NCD** prikazuje je u stvari stablo kataloga. Sada nije potrebno kucati duga naziva kataloga za izbor tekućeg: dovoljno je tasterima sa strelicama pomeriti kursor i tako označiti katalog koji želimo da proglasimo za tekući. Kada se pomeri kursor (u stvari, osvetljeni naziv) treba pritisnuti taster **<Enter>**. Efekat je potpuno isti kao da smo upotrebili komandu **CD**. Znači, uvek kada vam zatreba da izmenite tekući katalog, možete koristiti program **NCD**, i zaberite željeni katalog i pritisnite **<Enter>**. Naravno, za to je potrebno imati instaliran program **NCD** na disku.



6. Program NCD za rad sa katalozima

Program **NCD** ima još mogućnosti, od kojih možete koristiti zamenu za komande **MD** i **RD**. Pritiskom na taster **<F7>** kreirate novi katalog koji je direktni naslednik tekućeg, a sa **<F8>** brišete tekući katalog (naravno, ako u njemu nema datoteka). Pošto program **NCD** vodi svoju internu evidenciju o katalozima na disku, dobro je koristiti ili *DOS* komande **MD** i **RD** ili program **NCD** za kreiranje i brisanje kataloga. Ako koristite i jedno i drugo, može se desiti da **NCD** „izgubi" evidenciju o katalozima. Tada treba upotrebiti komandu **<F2>** – **Rescan** koja ponovo analizira sve kataloge na disku.

DISKOVI

Sve opisane operacije sa katalozima i datotekama se obavljaju na disku. Svaki disk ima svoje ime u *MS-DOS*-u. Obično je to ima **C:**. Da li se razlikovali katalozi na raznim diskovima (kasnije ćemo videti da ih može biti više na istom računaru), na puno ime kataloga se na početak dodaje i naziv diska. Tako je puno ime kataloga iz prethodnog primera **C:\TEKSTOVI\PROBA**. Sasvim slično se radi i sa datotekama: puno ime datoteke se dobija kada se na puno ime kataloga doda naziv datoteke.

Vaš računar verovatno ima najmanje jedan disk. Danas diskovi mogu biti veoma veliki, pa je uobičajeno da se dele na više delova. Tu operaciju podržava i *MS-DOS*, pa vi kao korisnik imate utisak da radite sa nekoliko diskova, a u stvari radite sa delovima (particijama) jednog istog. Ostale particije (ili diskovi) obično se zovu **D:**, **E:**, itd. Kao i kod kataloga, uvek se jedan od diskova bira za tekući i sve naredne komande se odnose na njega. Svaki od ovih diskova se sasvim slično ponaša: svaki ima svoje stablo kataloga i u svakom katalogu su neke datoteke. Rad je potpuno analogan.

Tekući disk se menja jednostavnim kucanjem njegovog naziva. Na primer, ako otkucate komandu (uslovno rečeno) **D:** tekući (radni) disk postaje **D:**. U zavisnosti od toga kako vam je podešen prompt, videćete i na njemu promenu: ako vam je prompt podešen da pokazuje naziv tekućeg kataloga, dobićete naziv tekućeg diska i kataloga na novoizabranom disku.

Vrlo je važno da svaki disk „pamti" svoj tekući katalog: ako radite na disku **C:** za katalogom **C:\DRAW**, a zatim pređete na disk **D:**, promenite katalog u **D:\TEST\CHK**, a zatim se vratite na disk **C:**, tekući katalog je ostao **C:\DRAW**. Naravno, ponovnim prelaskom na disk **D:** prelazite u katalog **D:\TEST\CHK**.

Naziva možete shvatiti kao svojevrsne adrese: ponekad se koristi skraćeni oblik, ponekad duži, sve u zavisnosti od toga šta je potrebno u datom trenutku. Na primer, ako se radi samo sa diskom **C:** nema potrebe da se navodi prefiks **C:** – naziv je potpuno određen i sa **\CRTANJE\DRAW\SLIKE**. Naravno, puni naziv niko da neće smatrati – skraćivanjem naziva jednostavno štedite u kucanju.

Treba reći da su i podaci na disketi organizovani na potpuno isti način kao i podaci na disku. U stvari, i disketa se ponaša kao jedan disk i ime naziv **A:** ili **B:**. Tako kucanjem **A:** tekući disk postaje disketa i onda tu možemo raditi sa katalozima i datotekama. Pošto je disketa mnogo sporija od pravog diska, operacije će biti znatno usporene.

Na računarima se može instalirati i tzv. **RAM disk**. To je disk sa potpuno istim osobinama kao i klasičan disk ili disketa: ima stablo kataloga, datoteka itd. Ovakav disk se kreira u operativnoj memoriji zbog bržeg rada. Naime, pošto je elektronska memorija daleko brža od običnog diska, rad sa ovakvim diskom je daleko brži. Obično se koristi za ubrzavanje rada programa koji puno rade sa podacima na disku.

RAD SA DATOTEKAMA

Kako smo već videli, svaki katalog sadrži neke datoteke. Kada komandom **CD** (ili nekim programom kao što je **NCD**) pređete u željeni katalog, spisak datoteka se dobija komandom **DIR**: na ekranu će se pojaviti sve datoteke koje pripadaju tom katalogu.

Svaka od tih datoteka ima neku funkciju: neke od njih, kao što su one sa ekstenzijama **EXE** i **COM**, sadrže izvršne programe, neke razne podatke potrebne za rad programa, a neke ćete i sami kreirati pišući neki tekst, crtajući slike i sl.

Datoteke se vremenom stvaraju na disku i tako zauzimaju sve veći i veći prostor. Zato treba suvišne datoteke brisati sa diska. Za brisanje datoteka koristi se komanda operativnog sistema **DEL**. Ova komanda ima jedan argument i to je naziv datoteke koja se briše. Da se podsetimo: naziv datoteke se dobija kada se na puno ime kataloga doda naziv datoteke. Na primer, ako želimo da izbrišemo datoteku **PROBA.TXT** u katalogu **\TXT\TEST** otkucaćemo

```
DEL \TXT\TEST\PROBA.TXT
```

Operativni sistem posle ovoga briše zadatu datoteku i ispisuje poruku o tome (**1 File(s) deleted**) ili ispisuje odgovarajuću poruku ako je prilikom brisanja došlo do nekih problema.

Međutim, dosta retko se koristi baš ovaj oblik **DEL** komande. Ako se nalazite u nekom katalogu, verovatno vam treba da izbrišete datoteku u pravo iz tog kataloga, a ne iz nekog drugog. Naravno, i datoteka iz tekućeg kataloga se može brisati na prethodni način – navođenjem punog naziva. Ali, za tim jednostavno nema potrebe jer se podrazumeva da se operacija izvodi prvenstveno sa datotekama u tekućem katalogu.

Na primer, neka je tekući katalog

C:\TXT\TEST. Ako želimo da u njemu izbrišemo datoteku **PROBA.TXT**, osim komande

```
DEL C:\TXT\TEST\PROBA.TXT
```

možemo kucati i samo

```
DEL PROBA.TXT
```

Pošto nije naveden pun naziv datoteke, operativni sistem traži datoteku u tekućem katalogu.

MS-DOS vam omogućuje i da promenite naziv neke datoteke. Na primer, napisali ste neki tekst i on je sačuvan u datoteci **PISMO.TXT**. Kasnije ste odlučili da sam naziv „PISMO" ne znači previše i odlučili ste da datoteku nazovete **PIS_PERI.TXT**. Za takvu operaciju koristi se komanda **REN** (od *rename*, promeni ime):

```
REN PISMO.TXT PIS_PERI.TXT
```

Posle ove komande *DOS* vas obaveštava da je naziv promenjen. Ako datoteka koja se zove **PIS_PERI.TXT** postoji od ranije, biće javljena greška: ne mogu postojati dve datoteke u istom katalogu koje imaju isti naziv.

Datoteka sadrže razne podatke: tekstove, programe, podatke o poslovanju, slike... Videti sadržaj datoteke nije baš tako jednostavno: svaki program kreira datoteke u nekom svom formatu i samo on je u stanju da dešifruje sadržaj takve datoteke. Na primer, neka slika kreirana programom za crtanje za vas je samo besmislen niz brojeva, dok program za crtanje na osnovu tih brojeva nepogrešivo crta sliku. Neke datoteke i maju sadržaj koji „ima smisla" i za vas: to su tzv. datoteke sa standardnim *ASCII* tekstovima. Takve datoteke obično imaju ekstenziju **TXT** ili **DOC**. Da bi videli sadržaj takve datoteke koristite komandu **TYPE** koja sadržaj datoteke ispisuje na ekranu. Na primer,

```
TYPE PISMO.TXT
```

ispisuje sadržaj datoteka **PISMO.TXT** na ekran. Možete slobodno probati da primenite naredbu **TYPE** i na druge datoteke, samo što nećete dobijati smislene rezultate.

KOPIRANJE DATOTEKA I SKRAĆIVANJE KOMANDI

Sledeća veoma važna komanda koja se često koristi je komanda za kopiranje datoteka – **COPY**. Ona omogućava da se kreira datoteka čiji je sadržaj potpuno identičan polaznoj. Ovo je korisno kada se, na primer, prave rezervne kopije datoteka ili treba promeniti samo adresu u već otkucanom pismu, a pri tom želite da sačuvate i originalno pismo.

COPY naredba ima veliki broj različitih mogućnosti za kopiranje. Najjednostavniji oblik se primenjuje kada treba kopirati samo jednu datoteku:

```
COPY C:\DRAW\PISMO.WP C:\RAZNO\PISMO.WP
```

Ova komanda kopira datoteku **PISMO.WP** u katalogu **\DRAW** na disku **C:** u katalog **\RAZNO** na istom disku. Komandom CD možete doći u katalog **\RAZNO** i proveriti **DIR** komandom da je tamo zaista kreirana kopija datoteke **PISMO.WP**. Pošto se kopiranje obavlja unutar istog (tekućeg) diska, nije potrebno navoditi naziv diska, već je dovoljno

```
COPY \DRAW\PISMO.WP \RAZNO\PISMO.WP
```

Kopija ne mora imati isti naziv kao originalna datoteka, umesto **PISMO.WP** može se navesti bilo koje ime. Ipak, ovakav slučaj je najčešći i zato je omogućeno da se to skraćeno piše: kada datoteka kopija ima isti naziv kao i polazna datoteka, onda nije potrebno navesti pun naziv kopije, već samo katalog:

```
COPY \DRAW\PISMO.WP \RAZNO
```

I ovo nije maksimalno skraćena komanda. Pošto se ovakve komande svakodnevno koriste, omogućeno je još kraće pisanje. Na primer, ako je katalog **u koji** se datoteka kopira tekući, onda uopšte nije potrebno navesti katalog u koji se kopira: ako nije naveden, podrazumeva se da je to tekući katalog:

```
COPY \DRAW\PISMO.WP
```

Sasvim slično važi i ako je polazni katalog tekući: i tada nije potrebno navoditi puno ime kataloga, već samo naziv datoteke:

```
COPY PISMO.WP \RAZNO
```

Naredba **COPY** se koristi i za kopiranje datoteka sa diskete na disk. I tada važe sva prethodna pravila, s tim što je neophodno navesti naziv diska (pošto je različit). Na primer, ako se kopira datoteka **PISMO.WP** sa diskete na disk **C:** u katalog **\RAZNO** treba otkucati

```
COPY A:PISMO.WP C:\RAZNO
```

Pošto se datoteka **PISMO.WP** nalazi u već izabranom katalogu na disketi (naravno, to pretpostavljamo), nije potrebno navoditi puno ime kataloga na disketi: dovoljno je samo ime datoteke. Ako je na disku **C:** aktivan katalog **\RAZNO**, onda je dovoljno

```
COPY A:PISMO.WP C:
```

I na kraju, ako je **C:** aktivan disk, onda ni to nije potrebno navoditi:

```
COPY A:PISMO.WP
```

Ova pravila ne važe samo za **COPY** naredbu nego i za ostale komande. Pogledajmo kao primer komandu **DIR**. Njome smo do sada pregledali datoteke u tekućem katalogu. U stvari, **DIR** komanda služi za pregledanje sadržaja **proizvoljnog** kataloga:

```
DIR C:\RAZNO
```

prikazuje na ekranu sadržaj kataloga **C:\RAZNO**. Sada primenjujemo pravila za skraćivanje pisanja: ako je aktivan disk **C:** i katalog **\RAZNO** to ne treba pisati, pa ostaje samo poznato

```
DIR
```

Možda vam se u prvom trenutku učine komplikovane sve te konvencije o skraćivanju naziva kataloga i datoteka, ali zaista nije tako. Kada budete počeli da radite vrlo brzo će vam sve to preći u „refleks" i skoro da nećete razmišljati o tome „šta sad treba skratiti". A za početak možete koristiti i pune nazive kataloga.

DŽOKER SIMBOLI

Ponekad je potrebno izvršiti obradu nad grupom datoteka. Na primer, potrebno je sa diskete iskopirati **sve** datoteka na disk. Takav zadatak je moguće uraditi i kopiranjem jedne po jedne datoteke, ali to može potrajati. *MS DOS* olakšava takve operacije upotrebom specijalnih simbola koji se obično nazivaju **džoker** simboli. To su simboli **?** i *****. Kada se zadaje argument neke komande, **?** zamenjuje proizvoljan simbol, a ***** proizvoljan niz simbola u nazivu datoteke. Evo kako se to koristi: u nazivu datoteke upotrebljavaju se ovi simboli i tako se umesto **jedne** datoteka specificira **grupa** datoteka. Na primer, ako se napiše **SLIKA.***, to označava sve datoteke koje imaju naziv **SLIKA** i proizvoljan niz simbola kao ekstenziju. Tako toj grupi pripadaju datoteke **SLIKA.WPG**, **SLIKA.TXT**, **SLIKA.WP**, itd. Ako neku operaciju treba uraditi sa grupom datoteka, onda se kao argument navodi ime koje specificira tu grupu. Na primer,

```
COPY \DRAW\SLIKA.* A:
```

kopira sve datoteke sa nazivom **SLIKA** i proizvoljnom ekstenzijom iz kataloga **C:\DRAW** na disketu. Ako je potrebno izbrisati sve datoteke iz tekućeg kataloga koje imaju ekstenziju **TMP**, to se radi sledećom komandom

```
DEL *.TMP
```

Ako je potrebno izbrisati sve datoteke iz tekućeg kataloga, onda se koristi sledeća komanda.

```
DEL *.*
```

što označava datoteke koje imaju proizvoljan naziv i proizvoljnu ekstenziju – znači sve datoteke u katalogu. Ova komanda može odjednom uništiti veliki količine programa, pa se zato pre brisanja vrši kontrola. U svakom slučaju, komandu **DEL** uvek upotrebljavajte sa krajnjim oprezom.

Džoker simbol **?** se koristi za zamenu jednog simbola u nazivu. Na primer **SLIKA?.WPG** označava datoteke **SLIKA1.WPG**, **SLIKA2.WPG**, **SLIKAX.WPG**, itd. Tako se kopiranje svih tekstova čiji naziv počinje sa **OBRAZAC** a završava proizvoljnim simbolom, može uraditi sledećom komandom

```
COPY OBRAZAC?.TXT A:
```

Džoker simboli se mogu i kombinovati.

```
DEL OBRAZAC?.*
```

briše sve datoteke čiji naziv počinje sa **OBRAZAC** i ima najviše 8 simbola (može imati i 7!), a ima proizvoljnu ekstenziju.

STARTOVANJE PROGRAMA

Programi se čuvaju u datotekama sa ekstenzijama **EXE** ili **COM**. To su takozvane **izvršne** datoteke. Da bi se startovao neki od ovih programa, treba jednostavno otkucati njegovo ime. Na primer, da bi se startovao program **TEST.EXE** u katalogu **\UTIL** na disku **C:**, treba otkucati

```
C:\UTIL\TEST.EXE
```

Kada se neki program startuje, on nadalje preuzima ulogu posrednika od operativnog sistema i vi komunicirate sa tim programom. To znači da komande operativnog sistema (**DIR**, **COPY**, ...) više ne važe – sada možete koristiti samo komande koje startovani program omogućava.

Neko univerzalno pravilo za komande programa ne postoje: svaki program koristi neke svoje specifične komande, onako kako se autorima programa činilo da je najbolje i najlogičnije. Učenje tih komandi je najpreporučljivije iz originalnog uputstva. Ipak, određena pravila postoje, jer se autori programa trude da komande učine što lakšim za učenje i zato se komande prave po nekoj logici. To znači da će tasteri **<BS>** i **<Del>** skoro uvek služiti nekom brisanju, da taster **<Esc>** uobičajeno služi za prekid tekuće operacije i sl. Naravno, ova pravila ne važe baš uvek, ali ipak važe u većini slučajeva.

Za startovanje programa važe ista pravila za skraćivanje, kao i za argumente komandi. Na primer, ako se startuje program koji je na aktivnom katalogu na disku **C:**, dovoljno je napisati

```
C:TEST.EXE
```

Ako je disk **C:** aktivan, onda ne treba ni to pisati, pa se program u tekućem katalogu startuje jednostavnim navođenjem imena. Nije potrebno navoditi ni ekstenziju (ona je za programe **EXE** ili **COM**), pa je dovoljno otkucati

TEST

Neki programi se moraju startovati baš na ovaj način, tj. iz tekućeg kataloga. To je zato što osim izvršnih (**EXE**) datoteka pojedini programi imaju i podatke neophodne za rad u datotekama u istom katalogu, pa ako ih po startovanju ne nađu, neće raditi.

Neke komande *DOS*-a su takođe obični programi: pređite u katalog **C:\DOS** i otkucajte **DIR**. Videćete datoteke **FORMAT.COM**, **XCOPY.EXE**, itd. koja su upravo komanda **MS-DOS**-a.

PODEŠAVANJE SISTEMA

Neke programe je potrebno startovati veoma često. Na primer, tekst procesor **WP.EXE** ćete startovati veoma često ako pišete puno tekstova. Ako bi bilo potrebno uvek preći na katalog na kome se nalazi program **WP.EXE** ili kucati pun naziv programa, bilo bi to veoma zamorno. A tek *DOS* komande: zamislite da se pre svake *DOS* komande mora otkucati **CD \DOS**! Zato *MS-DOS* omogućuje da se neki češće korišćeni programi startuju sa bilo kog kataloga. Za to služi *DOS* komanda **PATH**. Pomoću nje se zadaje spisak kataloga na kojima treba tražiti startovani program, u slučaju da se on ne nalazi u aktivnom katalogu. Na primer, može se napisati:

```
PATH C:\DOS;D:\UTIL\NORTON;C:\WP;
```

Kada ovo otkucate, *MS DOS* neće direktno reagovati. Ali, nadalje će se svi programi tražiti i u navedenim katalozima. To znači, ako se nalazite u katalogu **D:\TEKST** i otkucate **WP**, *DOS* prvo traži program **WP.EXE** u tekućem katalogu, zatim u **C:\DOS**, pa u **D:\UTIL\NORTON** i na kraju u **C:\WP**, gde se i nalazi. Time je omogućeno da se *WordPerfect* startuje sa bilo kog kataloga na disku. Ovo ima još jednu prednost: ne morate svoje tekstove mešati sa datotekama u katalogu **C:\WP**, već možete otvoriti posebne kataloge samo za vaše tekstove. Ako otkucate komandu **PATH** bez argumenata, na ekranu će se ispisati trenutni spisak kataloga za pretraživanje.

U prethodnom tekstu smo videli šta je to prompt: to je niz simbola kojim nas *DOS* obaveštava da je spreman da prihvati sledeću komandu. Taj niz simbola se može menjati komandom **PROMPT**. Na primer:

```
PROMPT DOS:
```

Posle ove komande operativni sistem će se javljati promptom **DOS:**. Nema nikakve prepreke da za prompt stavite bilo koji niz simbola, pa i nešto ovako:

```
PROMPT Da,gazda?
```

Ali, mnogo je korisnije u prompt uključiti i neke informacije o, na primer, tekućem katalogu. Da bi prompt ispisivao pun naziv tekućeg kataloga, treba izvršiti sledeću komandu:

```
PROMPT=$p$g
```

U ovoj komandi se koriste specijalna oznake za tekući katalog i simbol ">" (**$p** je oznaka za naziv kataloga, a **$g** oznaka za ">"). Posle ovoga ćete u svakom trenutku znati naziv tekućeg diska i kataloga. Na primer, u katalogu **C:\UTIL** prompt će biti **C:\UTIL>**

DOS ima još komandi za podešavanje sistema – neke od tih komandi zahteva sam *DOS*, a neke zahtevaju i programi. Bilo bi zamorno svaki put pri startovanju računara iznova vršiti razna podešavanja. Zato postoji datoteka **AUTOEXEC.BAT** – možete je naći na katalogu **C:**. Kada se računar uključi, on prvo izvrši sve komande navedene u ovoj datoteci i tek onda ispisuje prompt i očekuje komandu od korisnika. Na taj način se neke „stalne" komande izvršavaju automatski. Pregledajte komandom **TYPE** sadržaj vaše datoteke **AUTOEXEC.BAT**: svakako ćete naći neke poznate komande kao što su **PATH**, **PROMPT**, **CLS** i sl.

Još jedna datoteka ima sličnu ulogu: to je datoteka **CONFIG.SYS** na istom katalogu. Ona ne sadrži komande *DOS*-a kao **AUTOEXEC.BAT** već razne direktive za instalaciju operativnog sistema.

FORMATIRANJE

Diskete su napravljene od savitljive plastike koja je premazana tankim slojem specijalne emulzije osetljive na promenu magnetnog polja. Nad tom emulzijom se kreće glava koja podatke upisuje tako što menja magnetno polje, što se odražava na raspored čestica u emulziji. Čitanje upisanih podataka se odvija tako što čestice u emulziji indukuju magnetno polje na koje reaguje glava za čitanje i upis.

Disketa se pre korišćenja mora pripremiti za rad: neophodno je izvršiti tzv. **formatiranje**. Pri formatiranju se proverava emulzija na disketi, obeležavaju mesta koja nisu ispravna i formatiraju delovi koji će se koristiti za čuvanje informacija. Ako pokušate da koristite disketu koja nije formatirana, operativni sistem neće uspeti da pristupi podacima na njoj, već će javiti poruku **Drive not ready** ili nešto slično. Zato se nove diskete pre rada uvek formatiraju.

Formatiranje se izvodi komandom **FORMAT**. Na primer, ako vam se disketna jedinica na računaru zove **A:**, formatiranje diskete može početi sledećom komandom:

```
FORMAT A:
```

Posle toga se ispisuje poruka da stavite disketu i pritisnete neki taster. Kada pritisnete bilo koji taster, formatiranje počinje. U zavisnosti od verzije *MS-DOS*-a koji koristite, ispisivaće se različiti podaci (procenat diskete koji je formatiran ili podaci o broju formatiranih sektora). Na kraju se postavlja pitanje **Enter volume label** – svaka disketa može imati svoj naziv koji čini najviše 11 simbola. Ukoliko želite, možete otkucati neki naziv, a možete pritisnuti i samo taster **<Enter>**. Time je formatiranje završeno: na kraju se ispisuje poruka o veličini slobodnog prostora i veličini neispravnog dela u bajtovima. Zatim se postavlja kontrolno pitanje za formatiranje sledeće diskete – ukoliko želite da formatirate sledeću disketu, odgovorite potvrdno i nastavite postupak.

Posle ove operacije disketa je spremna za rad. Na njoj nema ničega i možete proizvoljno upisivati datoteke i kataloge. Formatiranje se obavlja samo jednom – kasnije je nepotrebno, čak i štetno, zato što se formatiranjem gube svi podaci na disketi. Ipak, ponekad se diskete formatira više puta, na primer, ako nastanu problemi sa podacima na njoj.

Opisani slučaj je najčešći i najjednostavniji: postoji više tipova disketa, i kako će tačno glasiti naredba za formatiranje na vašem računaru zavisi od tipa disketne jedinice koju imate, verzije operativnog sistema i na kraju od onoga šta želite da postignete. Standardni slučaj je da vaš računar ima disketnu jedinicu koja podržava diskete veličine 5.25" i kapaciteta 1.2MB. To su diskete obeležene sa **HD** (**High Density**). U ovom slučaju je komanda za formatiranje obično **FORMAT A:**. Veoma slične ovim disketama su diskete sa oznakom **DD** (**Double Density**), ali one imaju kapacitet od 360K, znači 4 puta manje. Ove diskete na disketnim jedinicama od 1.2M treba formatirati komandom **FORMAT A: /4**. Postoje još manje diskete (3.5") za koje važe slična pravila. Najverovatnije da vaš računar spada u opisana dva slučaja. Ipak, može se desiti da vaš računar podržava neke sasvim neobične formate, pa je najbolje da se raspitate ili da pročitate originalnu dokumentaciju, ako ste je dobili.

Disk radi na potpuno istim principima kao i disketna jedinica, pa je i za korišćenje diska potrebno prethodno formatiranje. To se radi komandom **FORMAT C: /S**, ali ovu operaciju treba raditi sa krajnjim oprezom: formatiranjem diske uništevaju se **svi** podaci na njemu. Obično se računar dobija sa već formatiranim diskom, tako da se ova operacija radi jednom ili nijednom.

DISKETE SA OPERATIVNIM SISTEMOM

Kada se računar uključi, u memoriji se još uvek ne nalazi operativni sistem – on se nalazi na disku i tek treba da se učita. Po uključivanju računara operativni sistem se učitava sa diske u operativnu memoriju. Operativni sistem se prvo traži na disketi. Ako nema diskete u disketnoj jedinici, traži se na disku i ako se tamo nađe učitava se u memoriju. To je razlog što za vreme uključivanja ne treba držati disketu u disketnoj jedinici: operativni sistem će biti tražen na disketi (što možete videti po kontrolnoj lampici na disketnoj jedinici), a ako tamo ne postoji, biće prijavljena greška **Non-system disk or disk error**. Rešenje je jednostavno: treba izvaditi disketu i ponovo resetovati računar.

Međutim, ponekad je potrebno operativni sistem učitati baš sa diskete, a ne sa diske. Na primer, disk je pokvaren ili na neki način oštećen. Tada nije moguće uopšte startovati računar bez diskete na kojoj se nalazi operativni sistem. Zato je potrebno uvek imati jednu disketu sa operativnim sistemom. Takvu disketu je jednostavno napraviti: jednostavno treba isformatirati disketu komandom **FORMAT A: /S**. Opcija **/S** označava da disketa treba da bude sistemska, tj. da na nju treba upisati operativni sistem.

Kada se ovakva disketa formatira, na nju će se upisati sve datoteke koja čine *MS DOS*: program **COMMAND.COM** i neke koja su nevidljive za običnu **DIR** komandu (prepoznaćete ih po smanjenom slobodnom prostoru koji pokazuje komanda **DIR**).

Kada ovako pripremljenu disketu stavite u disketnu jedinicu i resetujete računar, operativni sistem će se učitati sa diskete. Disk **C:** i programi na njemu su i dalje normalno dostupni. Da bi se normalno koristile sve komande *MS DOS*-a potrebno je još na disketu iskopirati programe (datoteke) koja čine komande *MS DOS*-a sa diska. Obično je to kompletan sadržaj kataloga **C:\DOS**.

Za disketu sa operativnim sistemom (često se takva disketa zove *boot* diskete) važe potpuno ista pravila kao za disk: možete kreirati datoteke **AUTOEXEC.BAT** i **CONFIG.SYS**, možete kreirati čitavo stablo kataloga itd. Naravno, ograničenje je slobodan prostor koji je daleko manji nego na disku.

OSTALE KOMANDE

Kao što je već rečeno, *MS DOS* ima još puno komandi koje vam omogućavaju razne operacije sa datotekama i operativnim sistemom. Međutim, čak i profesionalci dosta retko koriste takve komande: već se uglavnom sve svodi na korišćenje prethodno opisanih **COPY**, **DIR**, **DEL**. Neke od preostalih komandi su i ostaci sa starijih verzija *MS-DOS*-a, pa ih nije ni preporučljivo koristiti. Ipak, postoji još izvestan broj komandi koje ponekad mogu zatrebati. U svakom slučaju, nabavite neki priručnik koji sadrži spisak svih komandi *MS DOS*-a: kad vam zatreba neka komanda, lako ćete naći njen opis i način korišćenja. Sasvim lepo vam za to može poslužiti i umetak o *MS DOS* u objavljen nedavno u „Računarima".

CHKDSK je komanda koja proverava isprav

nost kataloga i datoteka na disku i po potrebi vrši ispravke. Ovu komandu je potrebno povremeno izvršavati, jer se mogu pojaviti neki blokovi na disku koji nisu ispravno povezani sa stablom kataloga. Ovo se obično javlja ukoliko je računar isključen za vreme rada nekog programa. Tada program nije imao vremena da sredi sve podatke sa kojima je radio, pa se javljaju greške na disku. Iz ovog razloga nikada ne treba gasiti računar kada radi neki program, uvek prethodno treba izaći u *DOS*. Povremena provera podataka na disku vrši sa komandom **CHKDSK C: /F**. Ovim se proverava disk C: – ako želite proveru nekog drugog, navedite njegovo ime. Ova komanda ispisuje podatke o slobodnoj operativnoj memoriji, slobodnom prostoru na disku, itd. Ako postoje greške biće ispisana sledeća poruka.

```
10 Lost clusters found in 123 chains
Convert lost chains to files (y/n)?
```

Na pitanje treba odgovoriti sa „N". Posle ove operacije svi blokovi su postavljeni na svoje mesto.

DATE i **TIME** komanda služe za postavljanje datuma i vremena na časovniku u računaru. Računar je opremljen časovnikom koji radi i kada je isključen iz struje. Tako bi vreme i datum treba lo da bude ispravno uvek kada uključite računar. Ponekad je potrebno menjati ovo vreme, tada jednostavno otkucajte **DATE** za promenu datuma ili **TIME** za promenu vremena. Na osnovu ovog časovnika se određuje vreme poslednje promene na datotekama koje vidite kada se zada komanda **DIR**, ako časovnik nije tačan, ni ove vrednosti neće biti ispravne.

DISKCOPY komanda služi za kopiranje disketa. Svaku običnu disketu možete iskopirati i tako što običnom **COPY** komandom iskopirate celo stablo kataloga sa originalne diskete ne disk, a zatim vratite na drugu disketu. Međutim, pri tome se ne kopiraju skrivene, sistemske datoteke, pa je bolje za ovo koristiti komandu **DISKCOPY**. Prvo se stavi originalna disketa i zada komanda **DISKCOPY A: A:** (ovo u slučaju da nemate dve disketne jedinice istog tipa). Kada se sadržaj diskete učite u memoriju dobićete poruku da promenite disketu. Tada treba staviti disketu na koju kopirate polaznu i pritisnuti <Enter>. Obe diskete moraju biti istog tipa.

WORDPERFECT – KAKO DA NAPIŠEM TEKST NA RAČUNARU

Računari se koriste u razne svrhe. Ipak, jedna od najčešćih upotreba je za pisanje tekstova. Tekst se na računaru neuporedivo lakše piše i štampa nego na najmodernijoj mašini za kucanje tekstova. U stvari, ogroman broj računara se i koristi isključivo kao moćna mašina za pisanje.

Postoje brojni programi namenjeni, kako se to kaže, **obradi teksta** na računaru. Ovi programi pokrivaju sve potrebe, počev od unosa teksta, preko raznih obrada, do pripreme tekste za štampu. Prosečnom korisniku uglavnom nisu potrebni svi ti programi namenjeni profesionalcima. Obično je potrebno otkucati nekoliko strana tekste, dodati neku ilustraciju, sve to srediti i na kraju odštampati. Vrhunski rezultat se dobija ako se svaki od ovih poslova radi specijalizovanim programom koji je namenjen baš tom poslu, ali se odlični rezultati mogu postići i upotrebom programa koji u sebi objedinjuju sve ove funkcije. Jedan od takvih programa je i tekst procesor *WordPerfect*.

WordPerfect je jedan od najpopularnijih i najprodavanijih tekst procesora na svetu. Pri tom treba imati u vidu da je na ovom tržištu konkurencija veoma oštra: postoji mnogo kvalitetnih programa koji rade ovaj posao. Ipak, *WordPerfect* se već dugo održava na vrhu.

WordPerfect nije program bez mane: postoje njegovi brojni nedostaci, ali i pored toga on spada u kvalitetne i pouzdane proizvode. U našoj zemlji se koriste i drugi tekst procesori: *Chi Writer*, *MS Word* i sl. Svaki od njih ima neke prednosti i mane i verovatno nećete previše pogrešiti ako uzmete bilo koji od njih. Ipak, trebalo bi da imate u vidu da će vaše potrebe sigurno rasti, ono što vas zadovoljava danas, neće vam biti dosta sutra. Zato je dobro da izaberete neki tekst procesor iz vrhunske klase, jer on sigurno ima ono što će vam zatrebati sutra.

WordPerfect u najviše zameraju glomaznost i to ne bez razloga: on omogućuje ogroman broj operacija sa tekstom koji radite i za svaku tu operaciju treba pritisnuti određeni taster ili kombinaciju tastera. Iz postojanja ogromnog broja funkcija proizilazi da *WordPerfect* ima i ogroman broj odgovarajućih testera koje budući korisnik mora da nauči. Kako svaki normalni korisnik teži da sa što manje truda dođe do rezulta ta, mnogi se u startu odriču „funkcija *WordPerfect*-a koje im nikad neće zatrebati" i za pisanje koriste neki od jednostavnijih tekst procesora. Ako se zanemari lični ukus svakog čoveka, ovo je svakako pogrešno razmišljanje. Naime, tačno je da *WordPerfect* ima ogroman broj funkcija koje treba naučiti, ali ne moraju se pamtiti odjednom sve funkcije koje ovaj program ima: za normalan rad je potrebno znati nekoliko osnovnih komandi i to **naorate** naučiti na svakom tekst procesoru, bez obzira na njegovu složenost. Kada vam kasnije zatreba neka funkcija, možete konsultovati uputstvo ili odličan sistem za pomoć koji će vas brzo uputiti na ono što tražite. U suštini, jedine razlike koje ćete uočiti u prvoj fazi učenja između „jednostavnih" i „komplikovanih" programa za obradu teksta su različite kombinacije tastera kojima se obavlja određena operacija. A da li ćete tekst snimati tasterom **<F10>** ili **<Alt-W>** i nije tako važno, zar ne?

Ovaj izbor krije još zamki: ako se opredelite za neki jednostavniji program, rizikujete da vam jednog trenutka zatreba ono što taj program nema. A prelazak na neki novi program tada može biti veoma mukotrpan. U svakom slučaju se odlučite za neki program koji koristi više ljudi oko vas, tako ćete moći da pitate nekog iskusnijeg za savet i nećete biti prepušteni sami sebi ako dođete u kritičnu situaciju. Takođe, poznatiji programi imaju bolju podršku: lakše možete naći knjige, nove fontove za vaš štampač itd.

Naredni tekst predstavlja uvod u neko minimalno znanje koje vam je potrebno da biste počeli da radite sa *WordPerfect*-om. Nemojte se obeshrabriti dužinom ovog teksta: pravi minimum je još manji od onoga što je napisano. Ovde je dato dosta više, tako da bi uz ovu knjižicu mogli sasvim efikasno da koristite *WordPerfect* i kad prestanete da budete „apsolutni početnik".

OSNOVNE OPERACIJE

Da bi počeli rad sa *WordPerfect*-om dovoljno je otkucati **WP** to je poziv ovog programa. Naravno, podrazumeva se da je pre toga izvršena

instalacija. Kada startujete program, za trenutak se na ekranu pojavljuje informacija o programu (naziv verzija i sl. – slika 7), a onda se pojavljuje radni ekran *WordPerfect*-a. Ovaj ekran, u stvari, predstavlja elektronsku verziju čistog lista papira na kome nameravate da otkucate tekst. U gornjem levom uglu se nalazi kursor on pokazuje baš kao i u *DOS*-u, gde će se pojaviti sledeći simbol koji otkucate. Poslednja linija na ekranu sadrži neke osnovne informacije naziv i broj teksta (može ih biti istovremeno 2), pozicija kursora i sl. Pošto niste otpočeli pisanje, tekst još uvek nema nikakvo ime.



7. Naslovni ekran *WordPerfect*-a

Pokušajte da otkucate neku rečenicu. Na primer

```
Ovo je test_
```

Kursor se ponaša kao i u *DOS*-u posle ispisivanja otkucanog simbola pomera se na levo. Taster za brisanje poslednjeg otkucanog simbola **<BS>** ponaša se kao i u *DOS*-u: kada se aktivira briše se simbol levo od kursora i kursor se pomera na njegovo mesto. Na primer, ako sada pritisnete **<BS>**, dobićete sledeći rezultat

```
Ovo je tes_
```

Ispravke nisu ograničene samo na poslednje otkucano slovo tasteri sa strelicama omogućavaju da pomerite kursor na proizvoljno mesto u tekstu i tamo napravite ispravku, kao što ćemo videti malo kasnije.

Kada pišete na mašini, potrebno je na kraju svakog reda pomerati „kolica" na početak sledećeg. Tu zamornu operaciju kod tekst procesora nije potrebno raditi, jer se obavlja automatski. Na primer, pokušajte da otkucate još jednu, dužu rečenicu. Kada kursor dođe do kraja reda, auto matski će se pomeriti na početak sledećeg. Tako ste vi oslobođeni jedne veoma zamorne obaveze.

Tasteri sa strelicama služe za pomeranje kur sora po tekstu tamo je omogućeno da sa vretimo na proizvoljno mesto u tekstu i tamo uradimo naku ispravku. Na primer, pokušajte da pomeri te kursor u sredinu prve rečenice (posle reči **je**)

```
Ovo je_test
```

i tamo pritisnite taster **<BS>**. Time ste izbrisali prvi simbol sa leve strane, a zav preostali tekst sa desne strane kursora se pomerio za jedno mesto levo, kako bi se popunila nastala „praznina"

```
Ovo j_test
```

Sasvim slično možete da ubacite neki tekst koji ste zaboravili da napišete. Na primer, ako sada otkucate „**e prvi**", ovaj tekst će se „umetnuti" na poziciju kursora.

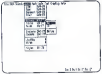
```
Ovo je prvi_test
```

Sav tekst sa desne strane kursora se sada pomera kako bi se napravilo mesto za novi tekst.

SNIMANJE I UČITAVANJE TEKSTA

Sa upravo napisanim tekstom možemo raditi razne stvari: štampati ga, menjati i sl. Jedna od najvažnijih operacija je **snimanje** teksta na disk u obliku datoteke. Time trajno memorišemo napisani tekst u datoteku na disku i omogućujemo da ga kasnije možemo opet menjati, štampati ili nešto treće. Da bi se napisani tekst snimio treba aktivirati taster **<F10>** koji aktivira funkciju zaduženu za snimanje teksta ili iz menija izabrati opciju **File/Save**. Kada se na jedan od ova dva načina aktivira snimanje, *WordPerfect* će postaviti pitanje **Document to be saved:** na šta treba otkucati naziv datoteke u koju će se upisati otkucani tekst. Ovaj naziv možete zadati sasvim proizvoljno, ali je dobro da on na neki način asocira na sadržaj teksta kasnije ćete na osnovu tog naslova morati da „pogađate" šta je unutra. Zato izbegavajte nazive poput **A.A** i sl. Ekstenziju izaberite i koristite je za sve tekstove napisane u *WordPerfect*-u. Na primer, datoteka se može zvati **PISMO.WP** na osnovu naziva znate da je reč o pismu, a na osnovu ekstenzije znate da je ovo tekst pisan u *WordPerfect*-u. Treba znati da snimljena datoteka nije „čitljiva" izvan *WordPerfect*-a osim teksta, u nju su upisani i razni kontrolni kodovi koje jedino *WordPerfect* može da protumači na pravi način. Zato ćete dobiti besmislen rezultat ako ovu datoteku pokušate da pregledate ili odštampate nekom komandom iz *DOS*-a, bez *WordPerfect*-a.

Komanda za snimanje datoteke je tipičan primer kako izgledaju komande *WordPerfect*-a: pritisneta određen taster i obavi se neka operacija. *WordPerfect* ima još jedan, alternativan način za aktiviranje svih funkcija izbor iz menija. Umesto da pritisnete **<F10>**, mogli ste da aktivirate meni pritiskom na taster **<Alt =>** (pritisnite taster **Alt** i držeći ga pritisnutog pritisnite taster sa oznakom "="; tada se pojavljuje meni ili pritiskom na levi taster miša. U oba slučaja se na vrhu ekrana pojavljuje padajući meni iz koga možete birati razne opcije tasterima sa strelicama (slika 8). Pritiskom na **<Enter>** birate željenu opciju, a ako ste se predomislili i ne želite da aktivirate ni jednu od opcija, pritisnite taster **<F1>** – on je zadužen za poništavanje prethodne akcije. Ovo je dosta važno, jer se često dešava da greškom aktivirate neku funkciju koju niste želeli.



8. Meniji u *WordPerfect*-u

Da li ste komandu aktivirali pritiskom na taster, ili izborom iz menija, sasvim je nevažno. Efekat u oba slučaja je potpuno isti, pa koristite ono što vam je zgodnije. Dok slabije poznajete komande, nije loše birati opcije iz menija, jer tamo imate spisak i lakše ćete naći željenu opciju. Kada se naviknete na to da se sa **<F10>** snima tekst, biće vam mnogo lakše i brže da odmah pritisnete **<F10>** nego da lutate po menijima. *WordPerfect* ima svoj naziv za svaku komandu (odnosno taster koji tu komandu aktivira). U daljem tekstu uglavnom će biti naveden i taj naziv. Na primer, komanda za snimanje datoteke naziva se **Save**.

Kada se tekst snimi komandom **<F10>**, na disku se formira datoteka koja sadrži taj tekst. Sve vreme dok ste kucali tekst on je bio u operativnoj memoriji – tek kada ste snimili tekst pravi se kopija teksta u memoriji i tako dobija datoteka na disku. Ako sada nastavite da kucate tekst, izmenićete samo primerak u memoriji datoteka na disku se ne menja. Tako se pojavljuju dve verzije teksta: jedna u memoriji (novija) i jedna na disku, starija. Ako ste u međuvremenu otkucali nešto korisno (dopisali još stranu ili dve teksta), logično je da želite da izbrišete staru verziju teksta na disku i sačuvate novu. Zato ponovo aktivirajte komandu **<F10>**. Pitanje o snimanju na disk se ponovo pojavljuje, samo što se sada nudi prethodno zadato ime datoteke. Ako želite da zamenite staru kopiju teksta u datoteci novom (iz memorije), jednostavno pritisnite **<Enter>**. *WordPerfect* tada pokušava da snimi tekst iz memorije u zadatu datoteku. Međutim, pošto iste novi da na disku već postoji datoteka sa istim imenom (stara verzija teksta), postaviće kontrolno pitanje: **Replace PROBA.TXT Yes(No)?** Ako želite da staru verziju zamenite novom, otkucajte **Y** (od **Yes**), a u suprotnom **N** – sada će vam ponovo biti postavljeno pitanje za naziv datoteke. Time se tekst iz memorije kopira preko stare verzije u datoteci i stara verzija se gubi.

Kada završite sa radom, potrebno je izaći iz *WordPerfect*-a. Za to se koristi taster **<F7>** ili opcija **Exit** iz menija. *WordPerfect* vas prvo pita da li želite da snimite tekst koji je trenutno u memoriji. Ako ste pre toga sa **<F10>** snimili tekst, nema potrebe da ponovo snimate. Ako niste, odgovorite sa **Y** – postupak je potpuno isti kao kada se aktivira **<F10>**. Zatim se postavlja pitanje **Exit WordPerfect Yes(No)?** Ukoliko završavate rad, pritisnite taster **Y**. Tako završavate rad sa *WordPerfect*-om i ponovo dobijate prompt *DOS*-a.

Taster **<F7>** je veoma važan u *WordPerfect*-u, to je tzv. taster **Exit** – njime se uvek izlazi iz i zabrane opcije, a i iz samog programa *WordPerfect*.

DOS komandom **DIR** možete proveriti da se u tekućem katalogu nalaze sve datoteke koje ste kreirali iz *WordPerfect*-a. Njih možete kopirati, brisati ili raditi bilo koju operaciju standardnim *DOS* komandama. Na primer, možete sve tekstove iskopirati na disketu i tako napraviti rezervnu kopiju važnog teksta, ako se disk pokvari, još uvek imate tekstove na disketama i obrnuto.

Osnovna prednost ovakvih „elektronskih" tekstova je ta što kasnije možete da ih menjate na mašinama za kucanje tako nešto se rešava isključivo raznim korektorima ili ponovnim kucanjem. Da biste tekst menjali, morate ga prvo učitati u *WordPerfect*. Kao prvo, komandom **CD** se pozicionirajte na katalog u kome je tekst koji želite da menjate. Zatim startujte *WordPerfect*. Kada se pojavi radni ekran, aktivirajte komandu za učitavanje teksta **<Shift F10>** (u meniju je to opcija **File/Retrieve**). Tada vam *WordPerfect* postavlja pitanje **Document to be retrieved**. Na to treba otkucati puno ime datoteke sa tekstom. Tekst se učitava i ponovo su vam na raspolaganju sve komande za rad sa tekstom. Kada završite izmene, ponovo snimite tekst sa **<F10>**. Ovaj postupak možete ponavljati proizvoljan broj puta.

Kada želite da menjate neki tekst, ne morate koristiti samo prethodni postupak za njegovo učitavanje, to je moguće uraditi na više načina. Na primer, moguće je pri startovanju *WordPerfect*-a navesti naziv datoteke koju treba učitati. Na primer, ako želimo da menjamo datoteku **TEST.WP**, *WordPerfect* se može startovati sa

```
C:> WP TEST.WP
```

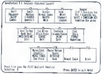
Kada ste već u *WordPerfect* u, postoji još jedan način (osim **<Shift-F10>**) za učitavanje i njega koristite kad niste sigurni kako se tačno zove datoteka. Tada treba pritisnuti **<F5>** (ili izabrati opciju **File/List Files** iz menija) i na pitanje **Dir:** otkucati ime kataloga u kome se nalazi tekst. Ako je tekst u tekućem katalogu pritisnite jednostavno **<Enter>**. Tada se dobija ekran kao na slici 9. To je radni ekran koji omogućava puno raznih operacija sa datotekama, ali za sada je dovoljno da se na ovom spisku pojavljuju i sve datoteke u tekućem katalogu i iz tog spiska možete izabrati onu koju želite da učitate. Potrebno je kursor pomeriti na naziv željene datoteke i iz menija sa dna ekrana izabrati **1-Retrieve** pritiskom na taster „1" ili na slovo „R". Posle toga se tekst normalno učitava kao komandom **<Shift F10>**. Ukoliko već postoji učitan tekst, postavice se pitanje **Retrieve into current document Yes(No)?** – *WordPerfect* ne zna da li želite da tekst sa diska učitate **unutar** (od nosno, da dodate na postojeći tekst) teksta koji je već u memoriji. Zato je pre učitavanja teksta potrebno „očistiti" memoriju: možete izaći i ponovo ući u *WordPerfect*, ali jednostavnije je da se otkuca **<F7>** i ona oba postavljena pitanja odgovorite sa "N" – time ste izbrisali tekst iz memorije i ostali ste u *WordPerfect*-u.



9. Rad sa datatekama

EDITOVANJE (MENJANJE) TEKSTA

Kada se učite tekst u *WordPerfect*, možete ga proizvoljno menjati. U prethodnom delu ste upoznali neke jednostavnije komande za menjanje teksta. *WordPerfect* ima veoma mnogo ovakvih komandi, ali njih nije potrebno sve odjednom naučiti: za početak će vam biti dovoljno da poznajete samo osnovne, a kasnije ćete učiti ono što vam zatreba. Postavlja se pitanje, kako uopšte naći neku komandu. Jedan od načina je prelistavanje priručnika, ali daleko je jednostavnije iskoristiti komandu koju *WordPerfect* ima namenjenu baš pružanju takvih usluga korisniku. Reč je o tzv komandi **Help** (pomoć) koja se dobija pritiskom na taster **<F3>**. Ova komanda se može koristiti na više načina. Kada se aktivira taster **<F3>**, dobija se kratko objašnjenje ovih načina.



10. Help sistem

Ako se ponovo pritisne **<F3>**, dobija se ekran kao na slici 10. To je crtež funkcijskih tastera sa tastature (obično se nalaze iznad centralnog dela tastature i obeleženi su sa **F1**, **F2**, ...) i naziva funkcija koje su im dodeljene. Sa strane je data legenda funkcija: za svaki taster postoja **Shift**, **Alt** i **Ctrl** kombinacije. Prepoznaćete na ovom crtežu tastere **Retrieve**, **Save** i **List** koje smo upoznali. Ova slika obično služi da vidite kom tasteru je dodeljena određena funkcija. Izlaz iz **Help** sistema je tasterom **<Enter>**.

Drugi način za korišćenje **Help** sistema je traženje opisa nekog tastera. Na primer, pritisnite **<F3>** i onda pritisnite taster **<F10>**. Tako dobijate objašnjenje komanda vezana za taster **<F10>** (snimanje teksta). Ovo koristite kada želite da vidite objašnjenje neke funkcije.

Treći način omogućava traženje komande po nazivu. Može vam se učiniti da traženje nepoznate komande po nazivu nema smisla, ali kada se malo naviknete na nazive osnovnih komandi, vrlo lako možete pogađati kako će se zvati komanda koja radi neku operaciju. Da bi aktivirali traženje po nazivu pritisnite **<F3>** i otkucajte prvo slovo komande koju tražite. Na ekranu će se pojaviti spisak svih komandi koje počinju zadatim slovom.

Help sistem ima još jednu lepu osobinu: kada aktivirate neku opciju, pritisak na **<F3>** ispisuje informaciju upravo o toj opciji.

Sve ove mogućnosti možete koristiti „skačući" iz jedne u drugu: kada pritisnete **<F3>**, možete pritiskom na "L" dobiti spisak komandi na "L", a zatim pritiskom na **<F3>** dobijete sliku funkcijskih tastera, itd. Izlaz iz **Help** sistema dodeljen je tasteru **<Enter>**, a ne uobičajenom tasteru **<F7>**.

Sama izmena teksta je dosta jednostavna: prvo se postavi kursor na mesto koje je potrebno izmeniti i zatim se izvrši izmena. Kursor se pomera tasterima sa strelicama za jednu poziciju gore, dole, levo i desno. Ovakav način pomeranja kursora je nezgodan ako je potrebno da kursor pređe veće rastojanje i zato postoji još komandi za „grublje" pomeranje kursora. Ako se tasteri sa strelicama levo i desno koriste u kombinaciji sa tasterom **<Alt>**, kursor se pomera po jednu reč levo ili desno. Taster **<End>** pomera kursor na kraj reda. Kursor se pomera na početak reda tako što se pritisne taster **<Home>**, a zatim se pritisne taster **<Left>**.

Tasterima **<PgUp>** i **<PgDn>** kursor se pomera za jednu stranu gore ili dole – pri tome je „strana" deo teksta koji će biti odštampan na jednom papiru, a ne sadržaj ekrana.

WordPerfect prelom strane označava linijom sastavljenom od simbola „=" kada napišete dovoljno linija, *WordPerfect* automatski ubacuje oznaku za prelazak na sledeću stranu. Ako kasnije izbrišete ili dodate tekst pre kraja strane, prelom se automatski pomera na odgovarajuće mesto. Ukoliko želite neki tekst da „nasilno" počnete na sledećoj strani, upotrebite taster **<Ctrl-Enter>**: time upisujete kod za prelazak na novu stranu.

Na kraju, tasterima **<+>** i **<->** na pomoćnoj tastaturi (pazite pri tom da indikator **Num Lock** bude isključen!) kursor se pomera sa ekran gore/dole. Pomeranje na početak teksta se postiže tako što se prvo pritisne taster **<Home>**, a zatim taster sa strelicom na gore. Slično se kombinacijom **<Home><Down>** kursor pomera na kraj teksta.

Tekst uvek možete menjati na dva načina: prvi način je da se novi tekst „umeće" između starog teksta, a drugi je tzv. **prekucavanje** – novi tekst se piše preko starog teksta. Tasterom **<Insert>** naizmenično menjate način rada (još se zove i **mod** rada). Pri tome se u statusnoj liniji vidi tekst **Typeover** ako je aktivno prekucavanje teksta.

SKRIVENI KODOVI

Tekst procesori se razlikuju od standardnih editora za unošenje teksta što na neki način omogućavaju da se osim teksta unose i oznake za formatiranje teksta. Tako u *WordPerfect* u može te označiti deo teksta pojačanim slovima (masna ili **bold** slova), možete odrediti kolika će biti desna margina pri štampanju itd. Ove oznake *WordPerfect* direktno uključuje u tekst, ali one su za korisnika nevidljive. Počećemo sa jednostavnim primerom. Pojačana slova se dobijaju pritiskom na taster **<F6>**. Pritisnite ovaj taster i otkucajte neki tekst. Tekst koji otkucate posle toga biće prikazan slovima jačeg intenziteta ili druge boje. Ponovni pritisak na taster **<F6>** završava rad sa bold slovima.



11. Skriveni kodovi

Na ekranu korisnik ne vidi nikakve specijalne oznake za deo teksta koji je napisan masnim slovima, ali te oznake postoje u tekstu. *WordPerfect* omogućava korisniku da vidi te skrivene oznake: pritisnite taster **<Alt F3>** i dobićete na ekranu nešto poput slike 11. Posle aktiviranja ove komande, ekran se deli na dva dela: tekst koji ste kucali prikazuje se u gornjem delu ekrana, a u donjem se prikazuje tekst sa skrivenim kodovima. Kao što se vidi na slici, početak i kraj bold teksta je ograničen oznakama **[BOLD]** i **[bold]**. Ako pokušate da pomerate kursor preko ovih oznaka, ponašanje će biti sasvim očekivano: kursor ove oznake tretira baš kao i bilo koji normalan simbol. Međutim, kad su ove oznake nevidljive, čini vam se da kursor „zapinje" i ne pomera se, bez obzira na pritisnut taster – to je upravo zbog skrivenih kodova.

Ako kursor pomerite između oznaka **[BOLD]** i **[bold]** i napišete neki tekst, sav taj tekst biće označen bold slovima. Možete i izbrisati ove oznake, čime sav označeni tekst pretvarate u normalan. Pri tome se automatski briše i druga oznaka. Ako su skriveni kodovi vidljivi (uključena je opcija **<Alt-F3>**), specijalne oznake se brišu baš kao i svi standardni simboli. Ako nije uključena ova opcija, *WordPerfect* pre brisanja postavlja kontrolno pitanje **Delete [BOLD] (Yes) No?** Ponovnim pritiskom na **<Alt F3>** vraćate se na „normalan" rad.

Komandu **<Alt-F3>** treba koristiti uvek kada počnu da vam se dešavaju „čudne" stvari: tekst se nekontrolisano pomera po ekranu, kursor ne reaguje na komande i sl. Pogledajte da li ste greškom u tekst ubacili neke skrivene kodove: obično brisanje otklanja problem.

Skriveni kodovi se ne koriste samo za masna, već za bilo koju drugu vrstu slova. Na primer, potpuno isto kao što sa tasterom **<F6>** dobijaju masna, tako se pritiskom na **<F8>** dobijaju podvučena slova (**Underline**). Sada će oznake za početak i kraj podvučenog teksta biti **[UND]** i **[und]**.

Skrivene oznake se koriste i za označavanje kraja reda i pasusa. Ranije smo videli da WordPerfect automatski prebacuje tekst u sledeći red kada kursor dođe do kraja reda. U stvari, tu se ubacuje oznaka [SRt] koja označava kraj reda. Ove oznake se automatski upisuju i brišu kada dodate ili izbrišete neki tekst time se dužina linije menja, pa WordPerfect ponovo vrši preformatiranje. Zbog ovoga ne treba pritiskom na taster <Enter> označavati kraj svakog reda – ovaj taster treba koristiti isključivo za oznaku kraja pasusa. Kada dođete do kraja pasusa, pritisak na <Enter> ubacuje kod [HRt]. Ovaj kod znači da korisnik traži prelazak u sledeći red i možete ga jedino vi izbrisati. Kada izbrišete ovu oznaku, tekst koji je bio u sledećoj liniji se pomera i nastavlja se rečenica u kojoj je kursor to je odgovor na pitanje „kako se u WordPerfect-u spaja tekst". Kada je potrebno razdvojiti tekst (podeliti jedan pasus na dva), dovoljno je pomariti kursor na željeno mesto i pritisnuti taster <Enter> – time se samo ubacuje jedan kod [HRt].

Skriveni kodovi se ne koriste samo za vrste slova i oznake kraja reda i pasusa, sve što zadate u vezi formata strane, izbora štampača, itd. rezultuje upisom najmanje jednog skrivenog koda u tekst. Pri tome treba znati da kodovi za formatiranje važe od mesta na kome se nalaze pa do sledećeg takvog koda ili kraja teksta. To znači, ako se na početku teksta upiše kod da je desna margina 25mm, tako će biti sve do kraja teksta ili dok ne upišete novu oznaku koja će pomeriti desnu marginu na neku drugu vrednost. Često se početnici žale da su promenili marginu, ali WordPerfect ne reaguje, nego zadržava staru vrednost. Rešenje je u inspekciji skrivenih kodova verovatno ste oznaku za novu marginu ubacili pre neke stare oznake koja poništava dejstvo nove. Zato komandom <Alt-F3> treba redovno kontrolisati skrivene kodove.

FORMATIRANJE TEKSTA

Smisao pisanja teksta na računaru je kasnije njegovo štampanje na štampaču. Da bi se tekst odštampao, prethodno treba podesiti razne parametre format papira na kome se štampa, margine, prored, itd. Ove parametre nije potrebno uvek iznova podešavati postoji način da se jednom podese i kasnije menjaju samo po potrebi.

Osnovno kod formatiranja teksta je izbor veličine strane i margina. Veličina strane je, u stvari, dimenzija papira na kome će se kasnije štampati tekst. Da biste zadali dimenzije (i još neke podatke) papira koji koristi vaš štampač, izaberite opciju Page iz menija Layout ili otkucajte <Shift-F8>. Meni koji se ispisuje omogućava vam da menjate razne parametre vezane za strani teksta. Za sada su nam važne opcije 5 i 7 podešavanje veličine papira i margina.

Podešavanje margina je jednostavno: pritisne se taster M ili cifra 5 i jednostavno se zada koliko će biti gornja i donja margina na strani. Koliko su margine može se pisati u raznim mernim jedinicama (to se može podesiti), ali vi možete bez obzira na to zadati margine u centimetrima tako što iza broja dodate slovo "c".

Na žalost, za podešavanje dimenzije papira na kome planirate da štampate tekst morate proći malo komplikovaniji put. Prvo se iz menija <Shift-F8> Page izabere sedma opcija Paper Size. Time se na ekranu dobije spisak od unapred definisanih veličina i karakteristika pošto jedan štampač može da koristi različite veličine papira, omogućeno je da napravite više definicija i da setim birate potrebnu. Ako je papir koji vama treba na spisku (na primer, unapred ste ga kreirali), jednostavno pomerite kursor na njega i pritisnite <Enter> – time ste izabrali dimenzije papira koji ćete koristiti. Ako prvi put koristite ovu opciju, najbolje je da sami kreirate svoju definiciju, nego da koristite neku unapred urađenu za koju niste sigurni tačno kakva je. Da bi kreirali svoju definiciju, izaberite opciju 2-Add WordPerfect vam onda nudi meni iz koga birate tip papira na kome želite da štampate tu možete ovim standardnog papira izabrati i nalepnice i sl. Za sada izaberite Standard. Posle toga dobijate sve informacije vezane za izbor papira tip, veličinu, kako se učitava papir i sl. Tip papira ste već izabrali i sad tamo stoji Standard. Ukoliko želite, možete promeniti veličinu papira izaberite opciju Paper Size i zatim označite papir A4 ili A3, u zavisnosti od štampača koji posedujete. U većini slučajeva je to A4. Zatim pritiskajte taster <F7> dok se ne nađete ponovo u tekstu.

Primetite da je na poziciji kursora ubačen skriveni kod koji u sebi sadrži definiciju koju ste izabrali potrebno je da kursor bude na početku teksta kad radite ovu operaciju da bi izbor važio za ceo tekst. Ukoliko se ovaj kod nađe negde u sredini teksta, važiće od tog mesta u tekstu pa do kraja, kako je već rečeno. Pošto se format papira na manja tako često u okviru jednog teksta potrebno je ovaj kod staviti na početak.

Da bi se konačno definisao oblik strane, potrebno je zadati još dva parametra levu i desnu marginu. To se, pomalo nelogično, nalazi u drugom meniju prvo treba pritisnuti taster <Shift-F8>, izabrati opciju Line iz menija Layout, a onda zadati kolika je leva, a kolika desna margina. Za zadavanje veličina u centimetrima važi ista napomena kao i za gornju i donju marginu. U meniju Line nalazi se još jedna važna vrednost razmak između linija (opcija Line Spacing). Tu možete zadati prored između redova. Uobičajeno je da se zadaje prored 1 (to će stajati ako ništa ne dirate), 2 (dvostruki prored) ali se može zadati i bilo koji decimalni broj kao prored, na primer 1.5.

Time smo definisali izgled strane WordPerfect sada ima podatke o tome koliki je papir na kome štampa, kolike su margine, pa samim tim i kolike su dimenzije teksta. Kada napišemo tekst, sve je spremno za završnu operaciju štampanje.

ŠTAMPANJE

Štampanje deluje kao veoma jednostavna operacija: uključi se štampač, ubaci papir, startuje se opcija programa koja štampa i sve je gotovo. Ali, stvarnost je daleko drugačija. Kada se radi sa mehaničkom naprevom, nestaje elektronska eleganciju i dešavaju se najprizemnije stvari: pa pir se zaglavi, kraći je od prethodno korišćenog itd. Autori WordPerfect-a su se dobro potrudili da rad sa štampačem učine što jednostavnijim, međutim, stvar po prirodi nije tako jednostavna.

Da bi se štampanje uspešno izvelo, prvo je potrebno WordPerfect-u saopštiti podatak koji tip štampača posedujete. WordPerfect podržava veoma veliki skup raznih štampača, pa ćete skoro sigurno u ovom dugačkom spisku pronaći i vaš štampač. Sve opcije vezane za štampanje nalaze se u jednom meniju do njega možete doći izborom opcije Print iz File menija ili pritiskom na taster <Shift-F7>. Dakle, prvo što treba urediti je da se izabere pravi tip štampača. Zato izaberite opciju Select Printer. Na ekranu će se pojaviti spisak štampača iz koga treba izabrati odgovarajući pritiskom na <Enter>. Ukoliko se desi da na ovom spisku ne pronađete vaš štampač, izaberite opciju Additional Printers. Iz ponuđenog spiska nađite vaš štampač. U ovom drugom slučaju biće vam postavljena dodatna pitanja, ali na njih uvek možete odgovoriti pritiskom na <Enter>. Time ste izabrali štampač koji ćete koristiti.

Kada želite da odštampate tekst koji ste uneli, izaberite opciju File/Print i iz ponuđenog menija izaberite Page. Time se strana na kojoj je trenutno kursor štampa na štampaču. Ako izaberete opciju Full Document, odštampaće se ceo tekst – od prve do poslednje strane. Kada aktivirate jednu od ove dva opcije, na ekranu ponovo dobijate tekst. Stvar je u tome što se štampanje odvija „u pozadini" dok WordPerfect štampa tekst, vi možete nastaviti se radom na istom ili nekom drugom tekstu.

Ako je WordPerfect podešen na rad sa pojedinačnim listovima papira, može se desiti da u dnu ekrana dobijete poruku Action required in Printer Control (potrebna je intervencija u maski za kontrolu štampača). WordPerfect ispisuje ovu poruku jer je potrebno pre štampanja svake strane pritisnuti taster "G" za početak štampanja. Svejedno, potrebno je ući u tzv. masku za kontrolu štampača.

Izaberite opciju Print iz menija File, a zatim izaberite Control Printer i dobićete na ekranu sliku 12. Na ovom ekranu su prikazane informacije o trenutnom stanju štampača koja strana se štampa (ako se nešto uopšte štampa), izabrane



12 Maska za kontrolu štampača

opcije pri tom štampanju i sl. Najvažnija su poruke ispisane posle Message i Action tu stoji poruka u čemu je problem, ako ima nekih problema sa štampanjem (nema papira, štampač ne reaguje, itd.). Ukoliko ste pri definisanju veličine papira izabrali da radite sa pojedinačnim listovima papira, pre štampanja svake strane dobićete poruku Press „G" to continue posle stavljanja papira u štampač treba pritisnuti slovo "G" da bi štampanje otpočelo. Kada se pojavi ova (ili neka slična poruka) može se desiti da radite nešto drugo, na primer, nastavili ste se kucanjem teksta. U tom slučaju će se u dnu ekrana pojaviti poruka Action required in Printer Control, što znači da je potrebno aktivirati masku za kontrolu štampača.

Pri štampanju se mogu desiti razne nepredvidive stvari: zadali ste da se odštampa ceo tekst, a niste stavili dovoljno papira za štampač ili ste ustanovili da to nije tekst koji ste želeli. Zato u meniju Control Printer postoje opcija Stop koja privremeno zaustavlja štampanje (nastavlja se izborom opcije Go) i opcija Cancel job koja trajno prekida štampanje.

Kada zadate nešto da se štampa, WordPerfect to kopira u poseban bafer i vraća vas u osnovni tekst, gde dalje možete menjati isti taj tekst koji se trenutno štampa. Međutim, ako za vreme štampanja izmenite tekst koji se upravo štampa, to neće promeniti tekst koji se štampa izmene će biti usvojene tek pri sledećem štampanju.

U opisima svih prethodnih komandi nedosta je dosta detalja. Naime, u nekim slučajevima WordPerfect postavlja dodatna pitanja o operaciji koju treba obaviti. Opis takvih situacija nije dat zato što ih ima prilično i nepotrebno bi iskomplikovale jednostavna situacije dovoljno je da znate osnovne principe, a detalje ćete lako naučiti. I pak, zato je potrebno da pažljivo pročitate svako pitanje koje vam WordPerfect postavlja nikako nemojte ne slepo kucati odgovore pre nego što dobro pročitate pitanje. Jedno od takvih pitanja se postavlja kada zadate nešto da se štampa a

onda pokušate da izađete iz *WordPerfect*-a. Tada će vam osim uobičajenih pitanja biti postavljeno i **Cancel all print jobs?**, tj. da li želite da poništite sve zahteve za štampanjem.

RAD SA TEKSTOM

U prethodnom delu smo upoznali neke osnovne komande za rad sa tekstom: kako se upisuje tekst, kako briše, kako se pomera kursor, i sl. *WordPerfect* poseduje još dodatnih komandi koje omogućavaju još jednostavniji i brži rad sa tekstom.

Često neku operaciju treba izvesti sa jednom ili više rečenica. Na primer, treba iskopirati rečenicu na drugo mesto, promeniti tip slova i sl. Za sve takve operacije koristi se takozvana **blok** komanda. Princip je isti za čitav skup operacija: pomoću blok komande markira se proizvoljan deo teksta, aktivira se komanda i tako se zadata operacija izvodi sa čitavim markiranim tekstom. Najjednostavniji primer je brisanje veće količine teksta. Naravno, to se može uraditi i korišćenjem „običnog" **<Del>** tastera, ali tako brisanje čitave strane može potrajati!

Početak blok operacije je pomeranje kursora na početak bloka teksta sa kojim želite da izvedete operaciju. Na primer, neka je na ekranu sledeća rečenica:

```
Ovo je probna rečenica.
```

Pomerite kursor na slovo **"j"** i pritisnite taster **<Alt F4>**. Iz menija ovu operaciju možete izabrati sa **Edit/Block**. Tada se u statusnoj liniji pokazuje poruka **Block on**, što označava da je uključeno markiranje teksta. Pomerite sada kursor bilo kojom od standardnih komandi za pomeranje. Tekst između slova **"J"** i tekuće pozicije kursora ostaje markiran. Kada tako označite blok teksta nad kojim želite da izvršite operaciju, pritisnite taster **<Del>**. Tada vam *WordPerfect* postavlja pitanje **Delete Block?** Ako odgovorite sa **Yes**, biće izbrisan čitav označen tekst.

Ako vam se desi da taster **<Alt F4>** aktivirate slučajno, ponovni pritisak na isti taster „ukida" označavanje. Primetite da kada aktivirate ovu komandu nije moguće normalno pisati tekst: pritisak na ma koji taster sa slovom pomera kursor na prvo sledeće pojavljivanje tog slova, što može biti korisno. Na primer, ako želite da označite čitavu rečenicu, dovoljno je pritisnuti "." da bi se kursor pomerio na kraj rečenice.

Sasvim slično se blok komanda koristi za izmenu tipa slova, na primer, ako želite da neku rečenicu koja je već napisana označite bold slovima, komandom **<Alt F4>** označite željenu rečenicu i pritisnite taster **<F6>**: sav markiran tekst biće konvertovan u bold. Potpuno istim postupkom se dobija podvučen tekst. Zapamtite dobro ovaj postupak: uvek kad budete želeli da neku operaciju uradite sa blokom teksta, primenjuje se opisani postupak: prvo se označi tekst, a zatim se aktivira željena operacija.

Markiranja teksta se koristi za dve izuzetno korisne operacije: kopiranje i pomeranje teksta. Kada želite da pomerite neki tekst na neko drugo mesto (na primer, greškom ste zamenili redosled dva pasusa), označite željeni tekst komandom **<Alt-F4>** i pritisnite **<Ctrl-Del>**. Označeni tekst se briše sa ekrana i upisuje se u specijalni bafer. U statusnoj liniji se pojavljuje poruka **Move cursor; press Enter to retrieve**: potrebno je pomeriti kursor na mesto gde želimo da pomerimo polazni tekst. Kada se to obavi, pritisak na **<Enter>** vraća tekst na poziciju kursora. Korišćenjem ove operacije možete slobodno menjati redosled rečenica, pasusa, pa i čitavih strana na veoma jednostavan način.

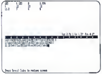
Slična operacija je **kopiranje teksta**. Ovde se označeni tekst ne pomera, već **kopira** na neko drugo mesto: originalni tekst ostaje neizmenjen. Posle označavanja teksta komandom **<Alt-F4>**, potrebno je otkucati **<Ctrl Ins>**, čime se označeni tekst kopira u interni bafer. Zatim treba pomeriti kursor na mesto gde je potrebno iskopirati polazni tekst i na kraju pritiskom na **<Enter>** sadržaj bafera se pojavljuje u tekstu.

IZBOR SLOVA I OBLIKOVANJE TEKSTA

Tekst se ne sastoji samo od rečenica koje se završavaju tačkom: tu su i razni naslovi, potpisi, tebele i sl. koji odudaraju od običnog teksta. Osnovna karakteristika ovih elemenata je da su pomereni na neku poziciju na strani: naslovi su centrirani, kolone tabela su jedna ispod druge, itd. Veoma je velika greška „poravnavati" tekst korišćenjem tastera **<Space>**: između svake dve reči treba kucati samo jedan prazan simbol, a *WordPerfect* tek pri štampanju poravnava desnu marginu i razmiče reči u tekstu ako je to potrebno. Znači, pri kucanju običnog teksta ne treba nikako praviti više od jednog praznog simbola između reči.

Naslovi se obično „uvlače" u odnosu na levu marginu. Ni za to ne treba koristiti taster **<Space>**, već taster **<Tab>**. Naime, *WordPerfect* razlikuje širinu svih slova. Tako je slovo **"m"** šire od slova **"l"**, pa će red u kome ime više slova **"i"** imati više simbola. Simbol za **space** (razmak) kod *WordPerfect*-a nema neku određenu širinu, već se to menja prema potrebi, kada se poravnava desna margina. Zato vam se može desiti da poravnate više redova tako da prvi simbol počinje uvek na istoj poziciji, a kad se tekst odštampa, ta linija bude veoma krivudava. Za poravnavanje teksta se koristi taster **<Tab>**: svaki pritisak na ovaj taster pomera kursor na određenu, fiksnu poziciju u liniji. Te pozicije možete i sami definisati izborom opcije **<Shift-F8>, Line, Tab** set, ali i unapred definisane vrednosti mogu sasvim lepo poslužiti. Dakle, ako na početku reda ubacite jedan **Tab** simbol (možete ga videti u skrivenim kodovima), svi takvi redovi će počinjati u jednoj „liniji". Ta linija je obično pomerena za nekoliko simbola od leve margine, što je idealno za naslove i početak pasusa.

Ukoliko pravite neku tabelu, tj. grupišete podatke po kolonama, možete i za to koristiti tab simbole. Naime, u okviru linije su definisane određena pozicije na koje se pomera kursor kod sledećeg pritiska na taster **<Tab>**. Raspored tih pozicija možete videti kada aktivirate pregled skrivenih kodova: na liniji koja razdvaja tekst i skrivena kodova nalaze se označene sve tab pozicije. Kada se pritisne taster **<Tab>**, kursor se pomera na prvu tab poziciju sa desne strane koja je uvek na istom mestu, bez obzira da li ste i koliko simbola otkucali pre aktiviranja tastera **<Tab>**. Tako se postiže da se razni „nazubljeni" podaci nađu jedni ispod drugih bez vaše (nepotrebne, a često i štetne!) intervencije (slika 13).



13. Formiranje kolona pomoću tabulatora

Naslovi se obično pomeraju na sredinu strana, između leve i desne margine. I ovde na treba koristiti „ručno" poravnavanje, već funkciju za centriranje teksta. Pre kucanja teksta koji želite da bude centriran, aktivirajte taster **<Shift-F6> (Centre)** i kursor će se pomeriti na sredinu linije. Tekst koji otkucate u toj liniji biće centriran. Na treba da vas brine ako tekst nije baš „strogo" centriran: tekst koji vidite na ekranu neće biti baš takav kad se odštampa. *WordPerfect* će se kasnije pobrinuti o tome da sve postavi na svoje mesto. Ako neki tekst naknadno zaželite da pomerite na centar, primenjuje se blok operacija: tekst označite komandom **<Alt-F4>**, a zatim pritisnite taster **<Shift-F6>**.

Slična funkcija je **Flush Right** (dobija se tasterom **<Alt-F6>**). Ona pomera tekst do desne margine – ovo je korisno kada želite da stavite tekst na kraj linije.

Ranije smo videli kako se menja tip slova tako da budu pojačana (**Bold**) ili podvučena (**Underline**). Ova dva tipa slova se najčešće koriste, ali nisu jedini koje *WordPerfect* podržava. *WordPerfect* vam omogućava da koristite sve tipove i oblike simbola koje podržava vaš štampač, i mnogo više. Svaka takva grupa simbola naziva se **font**. Vaš štampač podržava puno fontova koji se razlikuju po veličini, obliku i sl. Pošto takvih fontova može biti zaista mnogo, zamorno je za korisnika da stalno bira jedan od njih za svoj tekst: nego se izabere jedan font, a kasnije se samo menjaju razna osobine tog fonta: debljina, veličina, i sl. Na osnovu takvih zahteva korisnika *WordPerfect* pronalazi iz spiska fontova odgovarajući i automatski ga menja, tako da je korisnik pošteđen stalnog preturanja po dugačkim spiskovima. Osnovni font je uglavnom dobro izabran i možete ga koristiti bez izmene za većinu teksta.

WordPerfect podržava promenu veličine i tipa slova kojim se piše tekst. Obe ove stvari menjaju se izborom funkcije **Font** koja se dobija na taster **<Ctrl-F8>** ili izborom iz menija **Font**. Kada se pritisne taster **<Ctrl F8>**, za promenu veličine slova bira se opcija **Size**. Posle ovog izbora možete birati razna veličine slova: **Small, Large** itd. Na primer, ako izaberete **Large**, dobićete nešto krupnija slova od prethodno izabranih. Na ekranu nećete zaista videti uvećana slova, ali će i *WordPerfect* tako odštampati. Ako pogledate skrivene kodove, videćete da se na početku i kraju „uvećanog" teksta nalaze kodovi **[LARGE]** i **[large]**, sasvim slično kao kod bold teksta. Kada želite da prekinete sa pisanjem uvećanog teksta, možete izabrati opciju **Normal** iz menija koji se dobija sa **<Ctrl-F8>**, ili još jednostavnije, pomerite kursor preko skrivenog koda **[large]**. Na isti način koristite i atribute **Extra Large**, **Very Large**, **Small** i **Fine**. Opcije **Superscript** i **Subscript** se malo razlikuju: to su slova koja se koriste za izložioce i indekse. Na primer, ako želite da napišete **X²**, otkucaćete sledeće:

```
X<Ctrl F8>SP2<Ctrl-F8>N
```

Možda deluje pomalo komplikovano, ali ako probate, videćete da je u suštini veoma jednostavno.

PRETRAŽIVANJE I ZAMENA

Prednosti računara u odnosu na običnu mašinu za pisanje počinju da se pojavljuju tek kada tekst otkucate: ako je kucanje na računaru *malo* lakše nego na mašini, kasnije ispravke su na mašini skoro nemoguće, dok su na računaru krajnje jednostavne.

Jedan od primera za to je i naredba za pretraživanje teksta. Na primer, u nekom delu teksta

ste napisali neku reč i želite da nađete to mesto. Ako je tekst manji (strana ili dva) to nije problem, ali šta ako ste napisali tekst od 200-300 strana? Zato se koristi naredba **Search** (**<F2>**). Kada aktivirate ovu komandu, *WordPerfect* traži da unesete reč koju tražite. Otkucajte željenu reč slovo po slovo i na kraju pritisnite ponovo taster **<F2>** (veoma važno – ne smete za oznaku kraja koristiti taster **<Enter>**!). Posle toga počinje traženje reči od tekuće pozicije kursora do kraja teksta. Ako je tekst veliki, ova operacija može prilično potrajati. Ako se nađe reč, kursor će se pomeriti na nju (precizinje, iza njenog poslednjeg slova). Ako ne postoji takva reč u tekstu, biće ispisana poruka **" Not found "** u statusnoj liniji. Ako želite da nastavite traženje iste reči (ako ste je koristili na više mesta u tekstu, na primer), jednostavno pritisnite dva puta **<F2>** i traženje se nastavlja.

Izraz „reč" treba shvatiti uslovno: radi se prosto o nekom nizu simbola. Tako možete tražiti reč **„abeceda"** ali i **„je plat"** – znači, i delova reči, pa i čitave rečenice. Ako reč zadate malim slovima, biće nađena sva pojavljivanja, bez obzira na to kakvim slovima su pisana. Na primer, ako zadate traženje reči **„abc"** biće nađene reči **„abc"**, **„ABC"** i **„Abc"**. Ako zadate velika slova, biće nađene reči pisane isključivo velikim slovima. Pri ovome treba da vodite računa o našim slovima: *WordPerfect* obično naša slova tretira kao zagrade i obične simbole, u zavisnosti od načina instalacije.

Ova operacija može biti daleko korisnija nego što se čini na prvi pogled. Na primer, može poslužiti u otkrivanju nekih grešaka u tekstu koji ste napisali. Pravila koja se odnose na pisanje tekstova nalažu da ni jednu reč ne treba upotrebljavati prečesto u tekstu. Autor ne može uvek sveano da kontroliše ovo pravilo: ponekad se desi da izvesne reči ili konstrukcije prosto zatrpaju neki deo teksta. Ako pri ponovnom čitanju teksta posumnjate da se neka reč prečesto ponavlja, jednostavno aktivirajte komandu **Search** i moći ćete da pregledate sva mesta na kojima se pojavljuje ova reč. Tako možete lakše odrediti da li ćete možda neka pojavljivanja zameniti nekim drugim terminom.

Sličan primer je neka greška pri kucanju teksta. ako ste jednom napravili grešku pri kucanju neke reči, može se desiti da ste istu grešku napravili i u nekom drugom delu napisanog teksta. Komandom **Search** možete pretražiti kompleten tekst i otkloniti eventualne greške. Za ovakvu primenu mnogo je korisnija jedna druga komanda: **Replace** komanda za zamenu teksta (dobija se tasterom **<Alt F2>**). Ova komanda isto kao i **Search** traži pojavljivanje zadata reči, ali po pronalaženju zamenjuje nađenu reč nekom drugom. Kada aktivirate ovu komandu, prvo se postavlja pitanje

w/Confirm Yes(No)?

kojim se određuje akcija kada se nađe zadata reč. Ako na ovo pitanje odgovorite sa "Y", pre zamene svake nađene reči biće vam postavljeno kontrolno pitanje. U suprotnom će se bez kontrole izvršiti zamena u čitavom tekstu. Posle ovog pitanja treba zadati reč koja se traži. Kraj unosa kao i kod **Search** komanda označavate pritiskom na taster **<F2>**. Zatim na isti način treba zadati reč kojom treba zameniti traženu reč. Korišćenjem ova komanda možete, na primer, zameniti reč „Petar" u celom tekstu sa „Pera".

I ova mogućnost može imati veoma zgodnu praktičnu primenu. Na primer, ako ste uočili da ste pogrešno pisali jednu reč u celom tekstu, problem rešavate zamenom pogrešne konstrukcije ispravnom u celom tekstu. Još bolja primena su skraćenice: ako često u tekstu koristite neku frazu (na primer, „po sklopljenom ugovoru" i sl.), možete pri kucanju teksta značajno uštedeti vreme ako umesto fraze kucate neku skraćenicu, a zatim jednostavno zamenite skraćenicu u celom tekstu punom rečenicom.

MAKROI

Makroi obično spadaju u stvari koje nisu preporučljive za početnike. U slučaju *WordPerfect*-a je to zaista tako, pod uslovom da želite da ih koristite u punoj meri. Međutim, osnova makroa je veoma jednostavna i zaista je šteta odbaciti u startu jednu tako praktičnu alatku.

Makroi vam omogućuju da pojednostavite rad tako što će *WordPerfect* umesto vas raditi neke poslove koji se ponavljaju. Na primer, ako vam je često potrebno da pišete neku frazu, definisacete makro koji to radi i ubuduće ćete, umesto da sami pišete nekoliko rečenica, jednostavno startovati definisani makro. U stvari, makroi su način da *WordPerfect* naučite kako da radi neke stvari.

Navešćemo jedan primer. Ako pišete poslovna pisma, verovatno ih završavate frazom „sa poštovanjem, Pera Perić". Definisaćemo makro koji ova radi umesto vas. Svaki makro je, u stvari, niz tastera koji vi zadajete. Da bi se koristio makro, prethodno se mora definisati. Za to se koristi taster **Macro Define** (**<Ctrl-F10>**). Kada se pritisne ovaj taster, *WordPerfect* vam postavlja pitanje **Define macro:** – potrebno je uneti naziv makroa koji definisete. Možete ukucati neki niz slova ili još jednostavnije, pritisnite neku **Alt** kombinaciju tastera. Na primer, pritisnite **<Alt P>**. Još je potrebno uneti opis makro (pitanje **Description**) i potom počinje definisanje makroa: svaki taster koji pritisnete biće upamćen i kasnije ponovljen. Dok se definiše makro, u statusnoj liniji treperi natpis **Macro def**, koji označava da je u toku definisanje makroa.

Otkucajte sada „sa poštovanjem, Pera Perić". Naravno, ako je kursor negde u sredini pisma, a ne na kraju, ovo može biti besmisleno. Zato pre početka definisanja makroa treba pomeriti kursor na kraj teksta. Upamtite, **svaki** taster koji pritisnete biće memorisan – znači i tasteri sa strelicama za pomeranje kursora. Kada završite sa kucanjem, pritisnite ponovo **<Ctrl-F10>** i time ste završili definisanje makroa.

Korišćenje već definisanih makroa je veoma jednostavno: ako ste za definisanje koristili **Alt** kombinaciju tastera (u našem primeru **<Alt-P>**), dovoljno je da pritisnete taj taster. Znači, ako sada pritisnete **<Alt-P>**, ponoviće se niz tastera koji je memorisan: na ekranu će se pojaviti „sa poštovanjem, Pera Perić". Ako ste kao naziv makroa naveli niz simbola, prvo treba aktivirati taster **<Alt-F10>** i napisati ime makroa. Efekat je u oba slučaja isti. Makroi sa **Alt** kombinacijama se obično koriste za neke poslove koji se često obavljaju (zbog jednostavnijeg pozivanja), dok su makroi sa nazivima zgodni kada se radi neka složenija obrada.

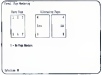
NUMERACIJA

Jedna od najzgodnijih osobina tekst procesora je da automatski numeriše stranice: o tome korisnik ne vodi nikakvu brigu. *WordPerfect* ne numeriše samo stranice, već i mnoge druge stvari: ilustracije, formule, tabele. U principu se sve svodi na isti princip: korisnik mora da zada način na koji će se vršiti numeracija, a *WordPerfect* radi ostatak posla.

Kod numerisanja strana važno je podesiti poziciju na strani gde će se ispisivati broj strane i format tog ispisa. Obe ove stvari se podešavaju izborom opcije **<Shift-F8>, Page, Page Numbering**. Posle ovog izbora podešavaju se razni parametri. Najvažniji su **Page Number Style** i **Page Number Position**.

Sa **Page Number Style** podešavate šta će se ispisivati kao broj strane. Inicijalno je to samo broj strane, obeležen sa „B (na tastaturi se dobija sa **<Ctrl B>**), ali se može upisati i čitav string, na primer, **"Strana broj: „B"**. Ako se ovako zada, prilikom numeracije strana biće ispisivan tekst **„Strana broj: 1"**, itd.

Potrebno je podesiti i mesto na strani gde će se ispisivati broj strane. Izaberite opciju **Page Number Position** i dobićete masku prikazanu na slici 14. Iz ponuđenog izbora jednostavno izaberite gde će se ispisivati broj strane. Obe opisane opcije upisuju kod u tekst, pa ako želite da izbor ima efekta na ceo tekst obavezno pomerite kursor na početak teksta.



14. Izbor pozicije za broj strane

Ako pišete neki duži tekst, pametno je pre pisanja učiniti nešto na organizaciji. To će se obavezno pre ili kasnije vratiti na neki način. Uzmimo jedan primer: pišete neki duži tekst koji sadrži naslove i podnaslove u nekoliko nivoa. Glavni naslovi biće numerisani brojevima počev od 1. Na primer:

1. NASLOV

Podnaslovi će biti uobičajeno numerisani tako što se na broj prethodnog nivoa doda broj tekućeg poglavlja. Na primer, šesti deo u poglavlju 2 može biti

2.6. PODNASLOV

itd. Ako pišete neko „razgranatije" delo, nije nimalo retko da imate i po nekoliko nivoa u dubinu, pa naslov tipa **2.1.2.3.4.** može da bude veoma čest!

Pretpostavimo sad da je potrebno da napravite neki izmenu: jedno poglavlje u sredini treba izbaciti. Samo izbacivanje je lako uraditi blok operacijama, ali šta sa numeracijom poglavlja? Svaki naslov posle izbačenog treba smanjiti za jedan, a to može biti veoma veliki posao. Zato *WordPerfect* omogućuje automatsku numeraciju poglavlja, slično kao i automatsku numeraciju strana.

Princip se sastoji u tome da korisnik ne piše broj poglavlja, već samo označava nivo tog naslova, a *WordPerfect* na to mesto ubacuje specijalni kod i kasnije automatski numeriše poglavlja. Za automatsku numeraciju pritisnite taster **<Shift-F5>** i izaberite opciju **Para Num**. Na pitanje **Paragraph level** upišite nivo naslova (**1** je prvi nivo, **1 1** drugi, itd.). Posle toga se ne ekranu pojavljuje broj koji odgovara tom nivou. Dobijeni broj nemojte pokušavati da menjate ručno: to je u stvari, specijalni kod koji možete samo celog izbrisati.

Sada nema nikakvog problema oko naknadne numeracije poglavlja, posle bilo kakve akcije sa tekstom *WordPerfect* automatski menja sve brojeve. Može se desiti da ovom komandom ne dobijate brojeve već se označavanje vrši slovima. Ako to želite da promenite, izaberite opciju **Define** iz menija koji se dobija sa **<Shift F5>**. Korišćenjem ovog menija možete izabrati jedan od neko

like načina numerisanja, a možete definisati i neki svoj način. Naravno, definisanje novog načina numeracije važi od pozicije kursora, pa zato pomerite kursor na početak teksta pre aktiviranja ove opcije.

STILOVI

Rešavanjem numeracije poglavlja problemi sa naslovima ne moraju biti gotovi: bilo koja operacija koju treba uraditi sa **svim** naslovima može biti izuzetno komplikovana, ponekad i nemoguća. Na primer, uvek ste naslove prvog nivoa razdvajali od prethodnog teksta sa dva znaka **[HRt]**, koristili ste automatsku numeraciju i naslov je bio označen bold slovima. Na primer, naslov je bio ovakav kada se pogledaju skriveni kodovi:

```
[Hrt]
[Hrt]
[Par Num:1] [BOLD]Naslov[bold] [Hrt]
[Hrt]
```

Posle ovoga bi išao običan tekst koji čini poglavlje.

Ako je sad potrebno izmeniti nešto u naslovu prvog nivoa (slova treba da budu obična, a ne bold, na primer), to se mora uraditi sa svim naslovima prvog nivoa. Ručna izmena kod tole dužeg teksta ne dolazi u obzir. Jedno razumno rešenje bi bilo kreiranje makroa koji traži naslov prvog nivoa i briše oznake za bold. Ali, šta ako je ovakva izmena česta? Praviti makro uvek kada je potrebno napraviti izmenu u naslovu je veoma neracionalno. Problem otežava i to što nije sigurno da ste **baš** svuda upotrebili isti format za naslov: skoro je sigurno da ćete negde pri kraju teksta zaboraviti da li ste prethodno upotrebljavali dva ili tri prazna reda ispred naslova.

Prethodni problem se veoma prirodno rešava **stilovima**. Stil čini niz kodova koji koristite za određenu namenu u tekstu. Na primer, možete definisati stil kojim naslov prvog nivoa definišete kao u prethodnom primeru. Kasnije sve izmene izvodite samo na jednom mestu: u definiciji stila. Veoma prirodno i veoma moćno.

Pre nego što počnete pisanje većeg teksta, obavezno definišite stilove. Ne morate se previše zamarati time da napravite nešto savršeno: dovoljno je samo napraviti kostur, a kasnije izmene su ionako automatske. Prvi korak u definisanju stila je zadavanje kodova koji ga čine. Za to treba treba aktivirati taster **Style** (**<Alt F8>**) i izabrati opciju **Create**. U maski koja se pojavljuje treba uneti naziv stila (**Name**: treba uneti neki niz simbola po kome ćemo prepoznavati taj stil), za tip treba ostaviti **Paired**, i najvažnije, treba zadati kodove (opcija **Codes**). Kada se izabere ova opcija, dobija se ekran kao na slici 15.

Kod [Comment] označava tekst koji korisnik ukucava – znači, u našem primeru, naslov poglavlja. Sada jednostavno treba otkucati kodove koji će činiti naslov: ubaciti nekoliko **[HRt]** simbola ispred i iza tasterom **<Enter>**, ispred naslova ubaciti kod za numeraciju paragrafa, promeniti tip slova za naslov i izaći pritiskom na taster **<F7>**. Time je definisan novi stil. Potpuno isto se definišu i ostali stilovi potrebni za tekst. Mogu se koristiti ne samo za naslove i podnaslove, već za bilo koja mesta u tekstu koja je potrebno pisati po nekom pravilu.

Sada su stilovi spremni za upotrebu. Kada hoćete da napišete novi naslov, pritisnite taster **<Alt F8>**. Na ekranu će se pojaviti spisak definisanih stilova. Izaberite jedan pomeranjem kursora i pritiskom na taster **<Enter>**. Time se u tekst upisuju dva skrivena koda koji označavaju početak i kraj naslova, a kursor ostaje u sredini. Otkucajte tekst naslova: pojaviće se na istom mestu kao da su kodovi iz definicije stila iskopirani u tekst. Međutim, zasta nije tako: bez obzira što vi na ekranu vidite preskočene redove, broj naslova, itd., u tekstu se nalaze **samo** oznake za početak i kraj stila. Zato će brisanjem ovih kodova tekst naslova biti pomeren odmah uz poslednji simbol prethodnog teksta.

Kada završite sa pisanjem teksta naslova, strelicom na desno preskačete skriveni kod za kraj stila. Pri tome se kursor kreće malo „skokovito", jer preskače više simbola u normalnom tekstu.

Prava prednost stilova se vidi tek kada počnete sa izmenama: bilo kakva izmena na definiciji sed se jednostavno prenosi na kompletan tekst. Na primer, ako vam se učini da naslov treba pomeriti malo u desno, aktivirajte **<Alt F8>**, izaberite stil naslova, opciju **Edit** a zatim **Codes**. Pojavljuje se definicija u kojoj jednostavno dodajte jedan tab simbol ispred naslova i izađite se **<F7>**: time ste izvršili izmenu u **čitavom** tekstu.



15. Kreiranje stila

Sada nikakva operacija sa naslovima nije teška: centriranje, izbor slova, prebacivanje u različit nivo itd. sve se svodi na izmenu definicije stila.

GENERISANJE SADRŽAJA

Sigurno vam je jasno da je ručno pisanje sadržaja **veoma** težak posao. Kada se na to doda mogućnost izmene teksta, što se odražava na sadržaj, ispada da je ručno pravljenje sadržaja preporučljivo jedino u nuždi. *WordPerfect* omogućuje automatsko pravljenje sadržaja: na vama je da samo označite naslove i mesto gde želite sadržaj, a *WordPerfect* generiše sadržaj tako što redom „kupi" označene naslove i zajedno sa brojem stranice stavlja u sadržaj.

Prvo treba označiti mesto gde će se generisati sadržaj: pomeri se kursor na mesto gde treba postaviti sadržaj i aktivira se komanda **<Alt F5>**, opcija **Define**, a zatim **Table of Contents**. Zatim se unose podaci o sadržaju: broj nivoa naslova, da li se poslednji nivo prikazuje u „prelomljenom" formatu i kako treba prikazivati koji nivo. Opcije su uglavnom prihvatljive, pa za početak onesite samo broj nivoa. Time je u tekst ubačen skriveni kod koji označava mesto i parametre sadržaja.

Posle označavanja mesta za sadržaj treba označiti tekst koji će ući u sadržaj. *WordPerfect* nema načina da razlikuje običan tekst od teksta koji korisnik smatra naslovom, pa je potrebno redom definisati naslove korišćenjem komanda **Block** (**<Alt F4>**) i zatim **<Alt F5>**, opcijom **To C**. Još je potrebno da unesete nivo naslova i time je označeni deo taksta definisan kao naslov koji ulazi u sadržaj.

Treća faza je samo generisanje sadržaja, što se radi kada je takst završen: aktivirajte **<Alt-F5>**, zatim opciju **Generate** i na kraju **Generate Tables, indexes,.**. Posle ovoga počinje generisanje sadržaja, što može da potraje kod većeg teksta.

Prethodni posao može značajno da se skrati ako koristite stilove za naslove. Sada nije potrebno juriti naslove po celom tekstu i pojedinačno ih označavati, već je dovoljno u definiciji stila tekst naslova označiti komandom **<Alt F5>**. Time je posao sveden samo na definisanje sadržaja i generisanje, što je izuzetna ušteda.

PODEŠAVANJE PROGRAMA

Pre korišćenja bilo kog programa potrebno je izvršiti **instalaciju**: to je postupak u toku koga se program i njegovi podaci kopiraju sa disketa na disk. Osim kopiranja podataka, prilikom instalacije vrši se podešavanje raznih parametara rada programa. Obično svaki komercijalni program ima poseban program koji upravlja instalacijom i omogućava vam da interaktivno podesite razne parametre.

I *WordPerfect* ima takav program za instalaciju: na prvoj instalacionoj disketi koju dobijete nalazi se program **INSTALL** koji je zadužen za instalaciju *WordPerfect*-a. Instalacija je u suštini prilično jednostavna: program vas vodi kroz sistem menija i pitanja, tako da jednostavnim odgovorima o vašem računaru i štampaču postižete da se *WordPerfect* ispravno instalira. I pored toga, ova operacija je prilično osetljiva i zasta bi bilo najbolje da je za početak prepustite nekom ko ima više iskustva. Postoji mnogo detalja prilikom instalacije koje treba znati da bi se postiglo maksimum. Zato obavezno potražite pomoć prilikom instalacije ili pažljivo pročitajte umetak o *WordPerfect*-u iz starijih brojeva „Računara".

Kada se program instalira na disk, još uvek nije sve spremno za korišćenje – javlja se problem zbog naših slova **ŠĆČŽĐ**. Na žalost, rešenje ovog problema zavisi od mnogo faktora, pa bi vam dobar savet i ovde teškako bio dragocen. I pak, može se dosta toga i samostalno uraditi. Na svu sreću ovaj problem ćete samo jednom rešiti i kasnije možete zaboraviti na njega. Kada počnete da radite sa *WordPerfect*-om, mogu se desiti sledeći slučajevi:

– U vaš računar i štampač ugrađena su naša slova po YUSCII standardu (umesto simbola []_@\) i kada napišete tekst sa našim slovima, sve se korektno štampa na štampaču. Mogło bi se reći da ste u ovom slučaju najsrećniji – neko je rešio problem umesto vas i možete nastaviti normalno da radite.

– U vaš štampač su ugrađena naša slova, ali pri kucanju tih simbola na ekranu vidite zagrade ili neke slične simbole. U ovom slučaju video kartica vašeg računara najverovatnije nema naša slova. Ako imate *Hercules* karticu, jedini pravi lek je da ugradite u *EPROM* kartice naša slova. To se mora raditi specijalnim uređajem koji vi verovatno nemate: u ovom slučaju raspitajte se o ugradnji naših slova u video karticu. Ako imate VGA grafičku karticu, naša slova se mogu dobiti korišćenjem specijalnih programa. Takvi programi se dosta jednostavno nabavljaju, između ostalog, ima ih i na „Sezamu".

– Ako nemate naša slova ni u računaru ni u štampaču, problem u principu možete rešiti korišćenjem samo mogućnosti *WordPerfect*-a. Stvar je u tome što *WordPerfect* zvanično podržava veliki broj raznih simbola, u kojima su i naša slova. *WordPerfect* će ova slova i odštampati i prikazati na ekranu.

Ovakvi simboli se ne štampaju „crtaju", tj. štampaju se u grafičkom modu, pa takva slova „iskaču" po izgledu iz teksta i njihovo štampanje je mnogo sporije. Ako imate laserski štampač prethodna rečenica se uglavnom ne odnosi na vas, napomena važi samo za matrične štampače.

Na ekranu će se naša slova prikazivati ako imate VGA karticu: tada je jedino potrebno da po

deste odgovarajući tip kartice u *WordPerfect*-u. Startujte **WP**, zatim pritisnite komandu **<Shift-F1>**, izaberite opciju **Display**, zatim **Graphics Screen Type** i iz ponuđenog spiska izaberite vašu video karticu. Ako imate mono karticu, naša slova neće biti prikazana, ali će se prikazivati „najbliži" simbol. Na primer, umesto slova **Ž** prikazivaće se slovo **Z** i sl. U nedostatku boljeg, i ovo rešenje može poslužiti.

Osim podešavanja slova na ekranu i štampaču, morate podesiti i tastaturu tako da pritiskom na neke tastere dobijate naša slova. *WordPerfect* za sve simbole omogućava kucanje navođenjem njihovog koda. Pregledajte datoteku **CHARACTR.DOC**, u njoj se nalazi spisak svih simbola i odgovarajućih kodova koje podržava *WordPerfect*. Tamo ćete naći mnogo interesantnih simbola, od naših slova, kineskih do ćirilice. Svaki simbol ima svoj kod u obliku dva broja. Na primer, kod našeg slova **Š** je 1,176. Evo kodova i ostalih naših slova:

Š	**1,176**
š	**1,177**
Ć	**1,96**
ć	**1,97**
Č	**1,98**
č	**1,99**
Đ	**1,86**
đ	**1,79**
Ž	**1,206**
ž	**1,207**

Ovo se koristi u *WordPerfect*-u na sledeći način: pritisne se taster **<Ctrl-V>** i na pitanje **Key=** napiše 1,176. Na ekranu će se pojaviti naše slovo **Š**. U principu, možete uvek ovako postupati: uvek kada vam zatreba neko naše slovo, možete ga dobiti na ovaj način. Međutim, nije ni malo jednostavno za ova često potrebna slova pamtiti kodove i pritiskati po deset tastera za dobijanje jednog slova. Zato se koristi mogućnost *WordPerfect*-a da se predefiniše tastatura, tj. da se svakom tasteru dodeli proizvoljan simbol ili makro. Otkucajte **<Shift-F1>**, izaberite opciju **Keyboard**, zatim **Create** i otkucajte **YU** (to je naziv definicije tastature koju kreiramo). Potom izaberite opciju **Map**: na ekranu će se naći kompletan pregled tastature, svih tastera i kombinacija sa svim **Shift** tasterima. Sada je potrebno samo da dodelite naša slova tasterima koje ste odabrali: pomerite kursor na izabrani taster, pritisnite **<Enter>** i na pitanje **Key:** otkucajte **<Ctrl-V>** i kod tastera slova koje dodeljujete tom tasteru. Postupak ponovite za sva naša slova i zatim izadite sa **<F7>**. Time ste kreirali jednu definiciju tasture. Opcijom **Select** proglašavate tu definiciju važećom. Posle toga ćete imati naša slova i na tastaturi, i na ekranu i na štampaču.

Opisani postupak nije jedino rešenje ovog problema: postoji mnogo raznih uslova koji mogu učiniti da se vaš problem reši na neki efikasniji način. Zato obavezno prelistajte tekstove u ranijim „Računarima" o instalaciji *WordPerfect* a kada budete rešavali ovaj problem.